



Networking the world's business data™

SNMP

E/OS SNMP Support Manual

P/N 620-000131-620

REV A

Simplifying Storage Network Management

Record of Revisions and Updates

Revision	Date	Description
620-000131-000	6/2001	Initial release of Manual
620-000131-100	11/2001	Update to manual
620-000131-200	5/2002	Update to manual
620-000131-300	9/2002	Update to support EFCM 6.1 and 6.2
620-000131-400	10/2002	Update to support EFCM 6.1, 6.2, & 6.3
620-000131-500	2/2003	Update to support E/OS 5.1 and EFCM 7.0
620-000131-600	8/2003	Update to support E/OS 5.2 and EFCM 7.2
620-000131-610	11/2003	Update to support E/OS 6.0
620-000131-620	1/2005	Update to support E/OS 7.0

Copyright © 2005 McDATA Corporation. All rights reserved.

Printed January 2005

Ninth Edition

McDATA, the McDATA logo, McDATA Eclipse, Fabriccenter, HotCAT, Intrepid, Multi-Capable Storage Network Solutions, Networking the World's Business Data, nScale, nView, OPENready, SANavigator, SANpilot, SANtegrity, SANvergence, SecureConnect and Sphereon are trademarks or registered trademarks of McDATA Corporation. OEM and Reseller logos are the property of such parties and are reprinted with limited use permission. All other trademarks are the property of their respective companies. All specifications subject to change.

No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of McDATA Corporation.

The information contained in this document is subject to change without notice. McDATA Corporation assumes no responsibility for any errors that may appear.

All computer software programs, including but not limited to microcode, described in this document are furnished under a license, and may be used or copied only in accordance with the terms of such license. McDATA either owns or has the right to license the computer software programs described in this document. McDATA Corporation retains all rights, title and interest in the computer software programs.

McDATA Corporation makes no warranties, expressed or implied, by operation of law or otherwise, relating to this document, the products or the computer software programs described herein. McDATA CORPORATION DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. In no event shall McDATA Corporation be liable for (a) incidental, indirect, special, or consequential damages or (b) any damages whatsoever resulting from the loss of use, data or profits, arising out of this document, even if advised of the possibility of such damages.

Preface	vii
----------------------	-----

Chapter 1 Introduction to SNMP

SNMP Management	1-1
SNMP Simplified	1-2
SNMP Commands	1-2
Why Variables Exist In a Managed Device.....	1-3
How SNMP Changes Variables (Objects) in a Managed Device	1-3
Standard MIBs.....	1-4
Private Enterprise MIBs	1-5
Traps and Their Purpose	1-5

Chapter 2 McDATA SNMP Support

Overview.....	2-1
E/OS Trap Overview	2-3
E/OS Trap summary table	2-4
..... Enterprise-specific Port Status Change Trap	2-7
Enterprise-specific FRU Status Change Trap	2-7
Enterprise-specific Invalid Attachment Trap	2-8
Enterprise-specific Threshold Alert Trap.....	2-8
Enterprise-specific FRU Traps	2-8
Enterprise-specific Link Traps	2-10
FA MIB Switch Status Change Trap	2-10
FA MIB Event Trap.....	2-11
FA MIB Sensor Trap	2-12
FA MIB Port Status Change Trap	2-12
MIB Definitions: MIB-II.....	2-21

System Group.....	2-21
Interfaces Group	2-23
Address Translation Group	2-30
IP Group.....	2-31
IP Routing group	2-37
ICMP Group.....	2-43
TCP Group.....	2-48
UDP Group.....	2-53
SNMP Group.....	2-54
Fabric Element Management MIB.....	2-61
Fabric Element Management MIB Tables	2-61
Fibre Alliance MIB.....	2-96
Type definitions	2-96
Trap Types.....	2-146

Appendix A Fibre Alliance MIB

FA MIB	A-1
Textual conventions for this MIB	A-4
Connectivity unit group	A-6
Event group	A-34
SNMP trap registration group	A-65
Related traps.....	A-68
Conformance definitions	A-69
Conformance units	A-70

Appendix B FC Management MIB

FCMGMT-MIB Definitions	B-1
Connectivity unit group	B-4
Sensor table.....	B-20
Port Table.....	B-24
Event Group	B-37
Link Table	B-42
Port Statistics	B-49
FC Simple Name Server Table.....	B-67
SNMP Trap Registration Group	B-73
Related Traps.....	B-77

Appendix C McDATA Private Enterprise MIB

FCEOS.MIB	C-1
Textual conventions for this MIB	C-2
Enterprise Specific Object Identifiers.....	C-5

Fibre Channel product lines	C-5
Groups in FCEOS MIB	C-5
System Group	C-6
Fibre Channel FRU Group.....	C-10
Fibre Channel Port Group	C-13
NPIV Information	C-39
Fibre Channel Zoning Group.....	C-42
Fibre Channel Threshold Alert Group.....	C-47
FCEOS Enterprise-specific Trap Definitions	C-52

Appendix D SNMP Framework MIB

SNMP Framework MIB	D-1
Textual Conventions used in the SNMP Management Architecture	D-2
The snmpEngine Group	D-7

Appendix E MIB II

Groups in MIB II.....	E-1
System group	E-2
Interfaces group	E-5
Address Translation group.....	E-14
IP group	E-16
ICMP group	E-33
TCP group	E-40
SNMP group	E-50

Appendix F Fabric Element Management MIB

FCFE.MIB	F-1
Configuration group.....	F-6
Operation group.....	F-18
F_Port Fabric Login table.....	F-21
FxPort Fabric Login table.....	F-25
Error group	F-30
Accounting Groups.....	F-35
Class 2 Accounting table	F-40
Class 3 Accounting Group.....	F-43
Capability Group	F-46

Index	1
--------------------	----------

This publication is part of the documentation suite that supports the McDATA® Sphereon™ 3016 Fabric Switch, Sphereon 3032 Fabric Switch, Sphereon 3216 Fabric Switch, Sphereon 3232 Fabric Switch, Sphereon 4500 Fabric Switch, Sphereon 4300 Fabric Switch, ES-1000 Switch, ED-5000 Director, Intrepid® 6064 Director, and Intrepid 6140 Director.

Who Should Use This Manual

Use this publication if you are planning to use SNMP to manage any of the McDATA switching products listed above.

The publications listed in [Related Publications](#) provide considerable information about both concepts and McDATA products

Organization of This Manual

This publication is organized as follows:

[Chapter 1, *Introduction to SNMP*](#), provides an introduction and overview of Simple Network Management (SNMP) and its operation.

[Chapter 2, *McDATA SNMP Support*](#), describes specific information available through SNMP, especially the Management Information Bases (MIBs) that are supported and the SNMP traps generated by the McDATA directors and switches.

[Appendix A, *Fibre Alliance MIB*](#) lists the MIB definitions of Fibre Alliance MIB.

[Appendix B, *FC Management MIB*](#), lists the FC Management MIB 3.0.

[Appendix C, *McDATA Private Enterprise MIB*](#), lists the McDATA private enterprise MIBs.

[Appendix D, *SNMP Framework MIB*](#), lists the SNMP Framework MIB.

[Appendix E, *MIB II*](#) lists the MIB-II, the RFC1213.mib renamed.

[Appendix F, *Fabric Element Management MIB*](#) lists the definitions of managed objects for the Fabric Element in Fibre Channel Standard.

An [Index](#) is also provided.

Manual Updates

Check the McDATA web site at www.mcddata.com for possible updates or supplements to this manual.

Related Publications

Other publications that provide additional information about the products mentioned in this manual are:

- *McDATA Enterprise Fabric Connectivity Manager User Manual* (620-005001)
- *McDATA Products in a SAN Environment -Planning Manual* (620-000124)
- *McDATA ED-5000 Enterprise Fibre Channel Director Installation Manual* (620-005003)
- *McDATA ED-5000 Enterprise Fibre Channel Director Service Manual* (620-005004)
- *McDATA ED-5000 Enterprise Fibre Channel Director User Manual* (620-005002)
- *McDATA Intrepid 6064 Director Installation and Service Manual* (620-000108)
- *McDATA Intrepid 6140 and 6064 Director Product Manager User Manual* (620-000153)
- *McDATA Intrepid 6140 Director Installation and Service Manual* (620-000157)
- *McDATA SANpilot User Manual* (620-000160)
- *McDATA Sphereon 3016 and 3216 Fabric Switch Product Manager User Manual* (620-000151)

- *McDATA Sphereon 3032 and 3232 Fabric Switch Product Manager User Manual* (620-000152)
- *McDATA Sphereon 3016 and 3216 Switch Installation and Service Manual* (620-000154)
- *McDATA Sphereon 3032 and 3232 Switch Installation and Service Manual* (620-000155)
- *McDATA Sphereon 4500 Switch Installation and Service Manual* (620-000159)
- *McDATA Sphereon 4500 Switch Product Manager User Manual* (620-000158)
- *McDATA Sphereon 4300 Switch Installation and Service Manual* (620-000171)
- *Planning User Manual* (P/N 620-000220-010)
- *Event Management User Manual* (P/N 620-000224-010)
- *EFC Manager Software Upgrade Instructions Release 8.6* (P/N 958-000336-030)
- *SANavigator User Guide* (P/N 621-000013)

Manual Conventions

The following notational conventions are used in this document.

Convention	Meaning
<i>Italic</i>	Outside book references, names of user interface windows, panels, buttons, and dialog boxes
Bold	Keyboard keys
Click. As in “click the icon on the navigation control panel.”	Click with the left mouse button on the object to activate a function.
Right-click. As in “right click the product icon.”	Click with the right mouse button on the object to activate a function.
Select. As in “select the log entry.”	Click once on the object to highlight it.

Where to Get Help

For technical support, McDATA® end-user customers should call the phone number located on the service label attached to the front or rear of the hardware product.

McDATA’s “Best in Class” Solution Center provides a single point of contact for customers seeking help. The Solution Center will research, explore, and resolve inquiries or service requests regarding McDATA products and services. The Solution Center is staffed 24 hours a day, 7 days a week, including holidays.

NOTE: To expedite warranty entitlement, please have your product serial number available.

McDATA Corporation
380 Interlocken Crescent
Broomfield, CO 80021

Phone: (800) 752-4572 or (720) 558-3910
Fax: (720) 558-3851
E-mail: support@mcdata.com

NOTE: Customers who purchased the hardware product from a company other than McDATA should contact that company’s service representative for technical support.

**Forwarding
Publication
Comments**

We sincerely appreciate any comments about this publication. Did you find this manual easy or difficult to use? Did it lack necessary information? Were there any errors? Could its organization be improved?

Please send your comments via e-mail, our home page, or FAX. Identify the manual, and provide page numbers and details. Thank you.

E-mail: pubsmgr@mcdata.com
Home Page: http://www.mcdata.com
FAX: Technical Communications Manager
 (720) 558-8999

Ordering Publications

To order a paper copy of this manual, submit a purchase order as described in *Ordering McDATA Documentation Instructions*, which is found on McDATA’s web site, <http://www.mcdata.com>. To obtain documentation CD-ROMs, contact your sales representative.

Trademarks

The following terms, indicated by a registered trademark symbol (®) or trademark symbol (™) on first use in this publication, are trademarks of McDATA Corporation in the United States, other countries, or both:

<u>Registered Trademarks</u>	<u>Trademarks</u>
Fabricenter®	E/OS™
HotCAT®	Eclipse™
Intrepid®	Fibre Channel Director™
McDATA®	OPENconnectors™
OPENready®	SANvergence™
SANavigator®	Sphereon™
SANpilot®	
SANtegrity®	

All other trademarked terms, indicated by a registered trademark symbol (®) or trademark symbol (™) on first use in this publication, are trademarks of their respective owners in the United States, other countries, or both.

Introduction to SNMP

Network management is a broad term, including workstation configuration, assignment of IP addresses, network design, architecture, network security, and topologies. All this can fall within the scope of a network manager.

Any protocol for managing networks must allow virtually all network devices and systems to communicate statistics and status information to network management stations (network managers). This communication must be independent of the primary network transmission medium and impose little effect on the efficiency of the network. Network managers must be able to obtain status information from managed devices, and make changes in the way the managed devices handle network traffic. Simple Network Management Protocol (SNMP) is one way of meeting these requirements.

SNMP Management

SNMP is a mechanism for network management that is complete, but simple. It is designed on the manager/agent paradigm, with the agent residing in the managed device. Information is exchanged between agents (devices on the network being managed) and managers (devices on the network through which management is done).

There are many possible transactions between agents and managers. These transactions vary widely with the different types of devices that can be managed. With so many varied requirements for reporting and management, the list of commands a manager must be

able to issue is overwhelming, and every new manageable device can increase the list. SNMP was created to allow all these things to be easily done on any growing network.

SNMP operates on a simple fetch/store concept. With SNMP the available transactions between manager and agent are limited to a handful. The manager can request information from the agent or modify variables on the agent. The agent can respond to a request by sending information, or if enabled to do so, voluntarily notify the manager of a change of status on the agent (issue a trap).

With SNMP, administrators can manage the switch configuration, faults, performance, accounting, and security from remote SNMP management stations.

SNMP Simplified

SNMP is the only network management protocol that is widely available from many vendors of TCP/IP networks and internetworks.

SNMP:

- Allows network management with a simple set of commands.
- Allows new devices added to a network to be easily managed with minimal intervention.
- Is adequate for many basic network management needs.
- Is generalized for application to networks other than TCP/IP, such as IPX and OSI.
- Provides considerable versatility for managing a great many types of devices.
- Allows all networks to employ the same method for management.

SNMP Commands

A manager requests information from an agent by sending a single command, the Get command. The Get command also has a variation (GetNextRequest) that permits more efficient operation:

- GetRequest – Requests the value of a specified variable on the agent. This command is used to retrieve management data.
- GetNextRequest – Requests the value of the next variable after the one specified in the command. This command is used to retrieve lists and tables of management data.

An agent responds to a request by sending a single command, the `GetResponse` command. This command contains the requested information.

A manager changes information (variables) in the agent by sending a single command, the `SetRequest` command. This command is used to manipulate management data.

A trap is used by an agent to report extraordinary events. Refer to [Traps and Their Purpose](#) on page 1-5.

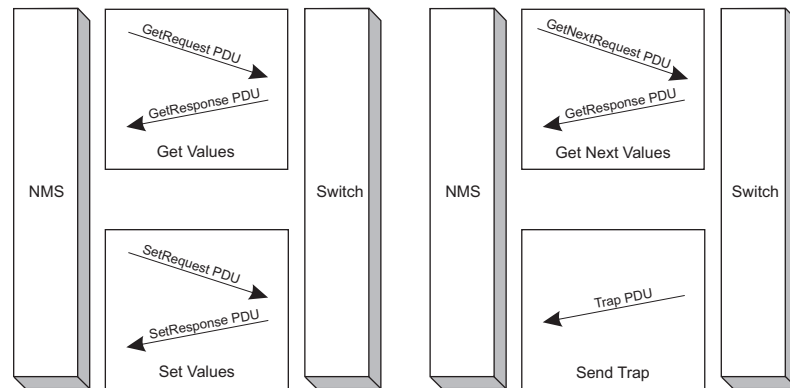


Figure 1-1 SNMP Commands and Responses

Why Variables Exist In a Managed Device

Variables are the means by which a switch or director (and other devices) keep track of their performance, control their own performance, and provide access to their performance for network managers. A simple example of a variable's use is to set a port offline and turn the port back on. Some variables just hold values that indicate status (for example error counts). SNMP allows the network managers to have access to some of the same variables for network management.

For purposes of the following explanation, an object is a data variable that represents an attribute of a managed device.

How SNMP Changes Variables (Objects) in a Managed Device

An agent is the entity that interfaces to the actual object being managed ([Figure 1-2](#) on page 1-4). The agent understands the language of SNMP and translates between the manager and the object. Objects may be retrieved and/or modified by the manager, and it is the agent's job to return the requested object's value. Within the agent is at least one, maybe several, collections of definitions

called Management Information Bases (MIBs). When an agent supports a standard MIB, it agrees to provide and make available the variables listed in the MIB.

A MIB is a hierarchical tree of groups and variables. Operators at a network management station enter a command with supported groups and variables from the MIB.

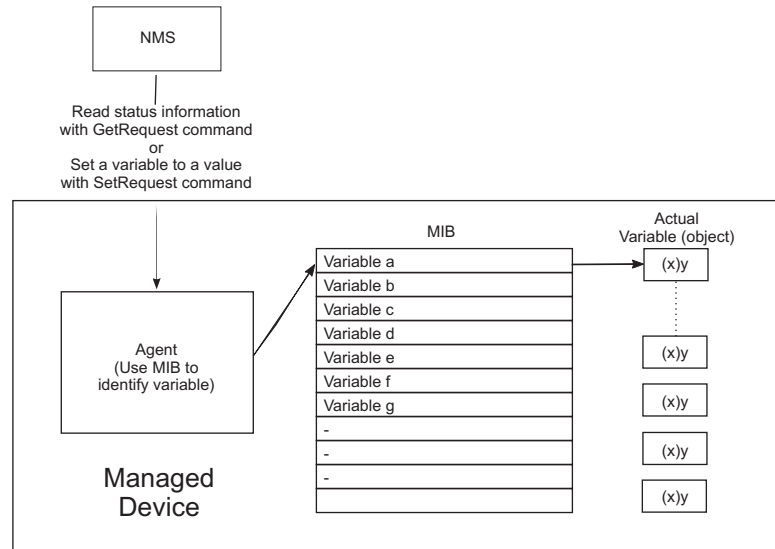


Figure 1-2 Retrieving or Setting Values Using MIBs

Standard MIBs

Standard MIBs are those created and approved by IETF and other Internet standards bodies and are readily available for use with SNMP network management stations. The standard MIBs provide a baseline of common operations across a wide variety of managed devices. Chapter 2 describes the standard MIBs used by the various McDATA products.

Standard MIBs supported by McDATA products are:

- MIB-II (Internet MIB) as described in RFC 1213: supported by all switches and directors.
- Fibre Alliance (FCMGMT) MIB, version 3.1: supported by EFC Server, Sphereon 4500 Fabric Switch, Sphereon™ 3216 Fabric Switch, Sphereon 3232 Fabric Switch, Intrepid™ 6064 Director, Intrepid 6140 Director, ES-3016, ES-3032, and ED-6064.

- Fibre Channel Fabric Element (FCFE), version 1.10: supported by all switches and directors.

Private Enterprise MIBs

Private MIBs are those provided by the manufacturer of the managed devices to allow management of device-specific items. Chapter 2 describes the McDATA private MIBs in more detail.

The McDATA private enterprise MIBs are:

- es1000 MIB, used by the ES-1000 switch
- ed5000 MIB, used by the ED-5000 director
- fcEos MIB, used by the Sphereon 4500 Fabric Switch, Sphereon 3232 Fabric Switch, Sphereon 3216 Fabric Switch, ES-3016 and ES-3032 switches, and the Intrepid 6064 and Intrepid 6140 directors (updated to support zoning, port binding, threshold alerts, and open trunking).

Traps and Their Purpose

Traps are unsolicited status reports, or status change indicators a managed object sends to a network manager. The destination address for traps is a configuration item for each managed agent.

Overview

SNMP is a protocol that uses the User Data Protocol (UDP) to exchange messages between an SNMP agent (in a managed device) and a management station residing on a network. Although SNMP can be made available over other protocols, McDATA only supports UDP.

To be monitored and managed remotely by a network management station, each switch or director is equipped with an SNMP agent. This agent is a software process within the switch that receives management requests and generates corresponding responses by accessing the data specified for the MIB-II, Fabric Element MIB, Fibre Alliance MIB, and FCEOS enterprise specific MIB. In addition, the agent gives each switch the ability to notify a management station when an important event occurs by sending a trap to the management station.

Five MIBs are supported:

- A subset of the Standard MIB-II for TCP/IP –based Internet as specified in RFC1213.
- Fabric Element MIB containing support for FL ports as specified in Fibre Channel standards.
- Fibre Alliance MIB (also referred to as the FC Management MIB).
- FCEOS MIB, the McDATA enterprise specific MIB supporting the McDATA switch and director firmware.

- SNMP Framework MIB.

NOTE: The remainder of this document refers only to the 3.1 version of the Fibre Alliance MIB, and uses its nomenclature. If you have need of information about the 3.0 version, refer to the MIB itself.

E/OS Trap Overview

NOTE: All E/OS traps are SNMPv1 format, regardless of MIB definition syntax.

SNMP traps are specific types of SNMP messages enclosed in UDP packets as shown:

[IP Packet [UDP Packet [SNMP Message]]]

The SNMP message format is:

[Version | Community | SNMP PDU]

There are different formats for the SNMP PDUs, including trap PDUs, for SNMPv1 and SNMPv2. These are summarized here:

SNMPv1 Trap PDU:

[Enterprise | Agent address | Generic trap type | Specific trap code | Time stamp | Object/Value 1 | Object/Value 2....]

The following descriptions summarize these fields:

Enterprise—Identifies the type of managed object generating the trap.

Agent address—Provides the address of the managed object generating the trap.

Generic trap type—Indicates one of a number of generic trap types.

Specific trap code—Indicates one of a number of specific trap codes.

Time stamp—Provides the amount of time that has elapsed between the last network reinitialization and generation of the trap.

Variable bindings—The data field of the SNMPv1 Trap PDU. Each variable binding associates a particular object instance with its current value

The following descriptions summarize the fields illustrated below for the SNMPv2 PDU format:

PDU type—Identifies the type of PDU transmitted (Get, GetNext, Inform, Response, Set, or Trap).

Request ID—Associates SNMP requests with responses.

Error status—Indicates one of a number of errors and error types. Only the response operation sets this field. Other operations set this field to zero.

Error index—Associates an error with a particular object instance. Only the response operation sets this field. Other operations set this field to zero.

Variable bindings—Serves as the data field of the SNMPv2 PDU. Each variable binding associates a particular object instance with its current value (with the exception of Get and GetNext requests, for which the value is ignored).

Get, GetNext, Inform, Response, Set, and Trap PDUs Contain the Same Fields:

[PDU type | Request ID | Error status | Error index | Object/Value 1 | Object/Value 2]

For the SNMPv2 trap pdu, the first and second variable bindings contain the uptime and the trap OID respectively. Following the uptime and trap OID are all the variable bindings specified in the MIB for that particular trap.

E/OS Trap summary table

This table shows the different kinds of traps supported by the switch E/OS firmware.

All E/OS traps are SNMPv1 format, regardless of MIB definition syntax.)

Trap	Severity	Sent because	MIB	Trap OID	E/OS	EFCM
Generic Authentication Failure	N/A	SNMP request from an invalid community is received	RFC-1157		YES	YES
Generic Link Up	N/A		RFC-1157		YES	NO
Generic Warm Start	N/A	Software reset	RFC-1157		YES	NO
Generic Cold Start	N/A	Power up	RFC-1157		YES	NO
ES port change	N/A	A change in port status	FCEOS	1.3.6.1.4.1.289.1	YES	NO

Trap	Severity	Sent because	MIB	Trap OID	E/OS	EFCM
ES FRU change	N/A	A change in FRU status	FCEOS	1.3.6.1.4.1.289.2	YES	NO
ES invalid attachment	N/A	Invalid attachment to a port.	FCEOS	1.3.6.1.4.1.289.3	YES	NO
ES threshold alert	N/A	Threshold specified in threshold table has been exceeded for a port.	FCEOS	1.3.6.1.4.1.289.4	YES	NO
ES FRU removed*	N/A	A FRU has been removed or transitioned to unknown status.	FCEOS	1.3.6.1.4.1.289.5	YES	NO
ES FRU active*	N/A	A FRU transitioned to the active state.	FCEOS	1.3.6.1.4.1.289.6	YES	NO
ES FRU backup*	N/A	A FRU transitioned to the backup state.	FCEOS	1.3.6.1.4.1.289.7	YES	NO
ES FRU update*	N/A	A FRU transitioned to the update/busy status.	FCEOS	1.3.6.1.4.1.289.8	YES	NO
ES FRU failed*	N/A	A FRU failed.	FCEOS	1.3.6.1.4.1.289.9	YES	NO
ES link bit error event*	N/A	The bit error rate for a link has exceeded an allowed threshold.	FCEOS	1.3.6.1.4.1.289.10	YES	NO
ES link no signal event*	N/A	Loss of signal or sync.	FCEOS	1.3.6.1.4.1.289.11	YES	NO
ES link NOS event*	N/A	A not operational primitive sequence timeout occurred.	FCEOS	1.3.6.1.4.1.289.12	YES	NO
ES link failure event*	N/A	A primitive sequence timeout occurred.	FCEOS	1.3.6.1.4.1.289.13	YES	NO
ES link invalid event*	N/A	An invalid primitive sequence is detected.	FCEOS	1.3.6.1.4.1.289.14	YES	NO
ES link added event*	N/A	A new link has been detected. NOTE: up to 10 seconds may elapse after link is added before trap is sent.	FCEOS	1.3.6.1.4.1.289.15	YES	NO

Trap	Severity	Sent because	MIB	Trap OID	E/OS	EFCM
Switch SCN	Alert	Change in switch status.	FC-MGMNT	1.3.6.1.2.1.8888.0.1	YES	YES
Switch Deletion	Alert	A switch is removed from management control.	FC-MGMNT	1.3.6.1.2.1.8888.0.2	NO	YES
Event SCN	Info	A new system event was generated.	FC-MGMNT	1.3.6.1.2.1.8888.0.3	YES	YES
Sensor SCN	Alert	Change in status for FAN/FAN2/POWER FRUs.	FC-MGMNT	1.3.6.1.2.1.8888.0.4	YES	YES
Port SCN	Alert	A change in port status.	FC-MGMNT	1.3.6.1.2.1.8888.0.5	YES	YES

* EOS 6.0 and later only.

The following sections describe each trap and the variables within the traps. For each variable, the OID is expressed as a numeric value first followed by a second line showing the symbolic object name.

Appended to the right of the OIDs are the index values for each object. Most of the objects within traps are actually table values. (That is, they refer to a MIB object which is part of a table defined in the MIB). Each SNMP table value must have an index appended to identify a specific table row.

For example, the enterprise specific port status change trap has the variable binding for `fcEosPortOpStatus`, which is a table entry value. So the OID for `fcEosPortOpStatus` (1.3.6.1.4.1.289.2.1.1.2.3.1.1.3) specifies a table column – to get a value for a specific port the table index (`port_number+1`) must be appended to the OID. If the trap occurred because of a change on port 5, then the actual variable OID would be 1.3.6.1.4.1.289.2.1.1.2.3.1.1.3.6.

Enterprise-specific Port Status Change Trap

This trap is sent for each port which has a status change. There is 1 variable binding as follows:

Binding	OIB	Value
1	1.3.6.1.4.1.289.2.1.1.2.3.1.1.3.port_number+1 fcEosPortOpStatus.port_number+1	New status value. See definition for fcEosPortOpStatus

Enterprise-specific FRU Status Change Trap

This trap is sent for each FRU which has a status change. There is 1 variable binding as follows:

Binding	OID	Values
1	<p>1.3.6.1.4.1.289.2.1.1.2.2.1.1.3.fru_code.fru_pos fcEosFruStatus.fru_code.fru_pos</p> <p>Where fru_code has one of the following values:</p> <ul style="list-style-type: none"> 0x01, Backplane 0x02, Control Processor card 0x03, Serial Crossbar 0x04, Shasta 32 center fan module 0x05, Fan module 0x06, Power supply module 0x07, Reserved 0x08, Longwave, Single-Mode, LC connector, 1 Gig (Port card) 0x09, Shortwave, Multi-Mode, LC connector, 1 Gig (Port card) 0x0A, Mixed, LC connector, 1 Gig (Port card) 0x0B, SFO pluggable, 1 Gig 0x0C, SFO pluggable, 2 Gig 0x0D, Longwave, Single-Mode, MT-RJ connector, 1 Gig 0x0E, Shortwave, Multi-Mode, MT-RJ connector, 1 Gig 0x0F, Mixed, MT-RJ connector, 1 Gig 0x10, F-Port, internal, 1 Gig <p>And where fru_pos is a number specific to each possible FRU position, which varies from product to product. For example, on a 6140 there are three fans numbered 1 to 3.</p>	New status value. See definition for fcEosFruStatus

Enterprise-specific Invalid Attachment Trap

This trap is sent when an invalid attachment occurs (a device is attached, with a WWN specifically disallowed by port binding). There is 1 variable binding.

Binding	OID	Value
1	1.3.6.1.4.1.289.2.1.1.2.4.1.1.4.port_number+1 fcEosPortAttachedWWN.port_number+1	WWN of invalid attached device. See definition for fcEosPortAttachedWWN.

Enterprise-specific Threshold Alert Trap

This trap is sent when port traffic exceeds a specified threshold. There are 2 variable bindings.

Binding	OID	Value
1	1.3.6.1.4.1.289.2.1.1.2.3.1.1.1.port_number+1 fcEosPortIndex.port_number+1	Port number of port with threshold alert.
2	1.3.6.1.4.1.289.2.1.1.2.6.1.1.1.threshold_number fcEosTAIndex.threshold_number	The index of the threshold which was triggered.

Enterprise-specific FRU Traps

The enterprise specific FRU traps (FRU removed, FRU active, FRU backup, FRU update, FRU failed: type codes 5-9) share the same bindings. There are 4 variable bindings for these traps:

Binding	OID	Value
1	1.3.6.1.4.1.289.2.1.1.2.2.1.1.1.fru_code.fru_position fcEosFruCode.fru_code.fru_position	The FRU code for this FRU. See table below.
2	1.3.6.1.4.1.289.2.1.1.2.2.1.1.2.fru_code.fru_position fcEosFruPosition.fru_code.fru_position	The FRU position for this FRU. The first position is 1.
3	1.3.6.1.4.1.289.2.1.1.2.1.15.0 fcEosSysSwitchName	The ASCII name of the switch
4	1.3.6.1.4.1.289.2.1.1.2.1.16.0 fcEosSysSwitchId	The Worldwide Name of the switch.

FRU Code	Description
1	Backplane
2	Control Processor card
3	Serial Crossbar
4	Shasta 32 center fan module
5	Fan module
6	Power supply module
7	Reserved
8	Longwave, Single-Mode, LC connector, 1 Gig
9	Shortwave, Multi-Mode, LC connector, 1 Gig
10	Mixed, LC connector, 1 Gig
11	SFO pluggable, 1 Gig
12	SFO pluggable, 2 Gig
13	Longwave, Single-Mode, MT-RJ connector, 1 Gig
14	Shortwave, Multi-Mode, MT-RJ connector, 1 Gig
15	Mixed, MT-RJ connector, 1 Gig
16	F-Port, internal, 1 Gig

FRU Code	Description
17	F-Port, internal, 1 Gig - XPM
18	F-Port, internal, 1 Gig - IPM

Enterprise-specific Link Traps

The enterprise specific link traps (link bit error, link no signal, link NOS, link failure, link invalid, link added: type codes 10 – 15) share the same bindings. There are 5 variable bindings for these traps:

Binding	OID	Value
1	1.3.6.1.4.1.289.2.1.1.2.3.1.1.1.port_index fcEosPortIndex.port_index	The fixed physical port number on the switch. It ranges from 1 to the number of physical ports that can be supported in the switch.
2	1.3.6.1.4.1.289.2.1.1.2.3.1.1.152.port_index fcEosPortName.port_index	A string describing the addressed port
3	1.3.6.1.4.1.289.2.1.1.2.3.1.1.153.port_index fcEosPortWWN.port_index	The Port WWN.
4	1.3.6.1.4.1.289.2.1.1.2.1.15.0 fcEosSysSwitchName	The ASCII name of the switch
5	1.3.6.1.4.1.289.2.1.1.2.1.16.0 fcEosSysSwitchId	The Worldwide Name of the switch.

FA MIB Switch Status Change Trap

This trap is sent when the switch status changes. There are 2 variable bindings.

Binding	OID	Value
1	1.3.6.1.2.1.8888.1.1.3.1.6.<unit-id> fcConnUnitStatus.<unit-id> Where unit-id is the WWN of the switch with 8 zeros appended for a total length of 16. Example: 1.2.3.4.5.6.7.8.0.0.0.0.0.0.0.0	Unit status. See definition for fcConnUnitStatus.

Binding	OID	Value
2	1.3.6.1.2.1.8888.1.1.3.1.5.<unit-id> fcConnUnitState.<unit-id> Where unit-id is the WWN of the switch with 8 zeros appended for a total length of 16. Example: 1.2.3.4.5.6.7.8.0.0.0.0.0.0.0	Unit state. See definition for fcConnUnitState.

FA MIB Event Trap

This trap is sent when an internal software event is generated. There are 4 variable bindings.

Binding	OID	Value
1	1.3.6.1.2.1.8888.1.1.3.1.1.<unit-id> fcConnUnitId.<unit-id> Where unit-id is the WWN of the switch with 8 zeros appended for a total length of 16. Example: 1.2.3.4.5.6.7.8.0.0.0.0.0.0.0	The value is the same as unit-id: the WWN of the switch with 8 zeros appended for a total length of 16. Example: 1.2.3.4.5.6.7.8.0.0.0.0.0.0.0
2	1.3.6.1.2.1.8888.1.1.7.1.5.<unit-id><event-index> fcConnUnitEventType.<unit-id><event-index> Where unit-id is the WWN of the switch with 8 zeros appended for a total length of 16. Example: 1.2.3.4.5.6.7.8.0.0.0.0.0.0.0 And where event-index is an integer index of the event table, a unique incrementing value assigned to each event. The event table always contains the most recent 200 events which met the filter criteria in place when the event occurred.	See definition for fcConnUnitEventType.
3	1.3.6.1.2.1.8888.1.1.7.1.6.<unit-id><event-index> fcConnUnitEventObject.<unit-id><event-index> Where unit-id is the WWN of the switch with 8 zeros appended for a total length of 16. Example: 1.2.3.4.5.6.7.8.0.0.0.0.0.0.0 And where event-index is an integer index of the event table, a unique incrementing value assigned to each event. The event table always contains the most recent 200 events which met the filter criteria in place when the event occurred.	The value of this variable is the OID for fcConnUnitId: 1.3.6.1.2.1.8888.1.1.3.1.1.<unit-id> Where unit-id is the WWN of the switch with 8 zeros appended for a total length of 16. Example: 1.2.3.4.5.6.7.8.0.0.0.0.0.0.0
4	1.3.6.1.2.1.8888.1.1.7.1.7.<unit-id><event-index> fcConnUnitEventDescr.<unit-id><event-index> Where unit-id is the WWN of the switch with 8 zeros appended for a total length of 16. Example: 1.2.3.4.5.6.7.8.0.0.0.0.0.0.0 And where event-index is an integer index of the event table, a unique incrementing value assigned to each event. The event table always contains the most recent 200 events which met the filter criteria in place when the event occurred.	Event description string with a maximum length of 80 characters. This string will contain a numeric event code and other values describing the specific event. See QMS-00019 for a description of events.

FA MIB Sensor Trap

This trap is generated whenever a status change occurs for a fan or power supply FRU. There is 1 variable binding.

Binding	OID	Value
1	1.3.6.1.2.1.8888.1.1.5.1.3.<unit-id>.<sensor-index> fcConnUnitSensorStatus.<unit-id>.<sensor-index> Where unit-id is the WWN of the switch with 8 zeros appended for a total length of 16. Example: 1.2.3.4.5.6.7.8.0.0.0.0.0.0.0 And where sensor-index refers to the FRU in the sensor table which has changed state. For example if sensor-index was 5, then you could look at the 5 th entry in the sensor table to determine which FRU was affected.	See description for fcConnUnitSensorStatus

FA MIB Port Status Change Trap

This trap occurs whenever a port status change occurs. There are 2 variable bindings.

Binding	OID	Value
1	1.3.6.1.2.1.8888.1.1.6.1.6. <unit-id>.<port-index> fcConnUnitPortStatus. <unit-id>.<port-index> Where port-index is the port number normalized to the range 1-140.	See definition for fcConnUnitPortStatus.
2	1.3.6.1.2.1.8888.1.1.6.1.5. <unit-id>.<port-index> fcConnUnitPortState. <unit-id>.<port-index> Where port-index is the port number normalized to the range 1-140.	See definition for fcConnUnitPortState.

Enterprise-specific Traps

fcEosPortScn

Type Number	1
Product Mapping	Generated when Fibre Channel port operational state changes.
Trap Variables	fcEosPortOpStatus

Description	An <code>fcEosPortScn(1)</code> is generated whenever a <code>Fc_Port</code> changes its operational state. For instance, the <code>Fc_Port</code> goes from online to offline.
-------------	---

fcEosFruScn

Type Number	2
Product Mapping	Generated when FRU operational state changes.
Trap Variables	<code>fcEosFruStatus</code>
Description	An <code>fcEosFruScn(2)</code> is generated whenever a FRU status changes to operational state.

fcEosPortBindingViolation

Type Number	3
Product Mapping	Generated when Port binding violation occurs.
Trap Variables	<code>fcEosPortAttachedWWN</code>
Description	An <code>fcEosPortBindingViolation(3)</code> is generated whenever the switch detects that a port binding violation occurs.

fcEosThresholdAlert

Type Number	4
Product Mapping	Generated when Threshold alert occurs.
Trap Variables	<code>fcEosPortIndex</code> <code>fcEosTAIndex</code>
Description	An <code>fcEosThresholdAlert(4)</code> is generated whenever a threshold alert occurs.

fcEosFruRemoved

Type Number	5
Product Mapping	Generated when a FRU is removed or its status changes to unknown.
Trap Variables	<code>fcEosFruCode</code> <code>fcEosFruPosition</code>

	fcEosSysSwitchName
	fcEosSysSwitchId
Description	An fcEosFruRemoved trap is generated when a FRU is removed or its status changes to unknown
<hr/>	
fcEosFruActive	
Type Number	6
Product Mapping	Generated when a FRU status changes to an active status.
Trap Variables	fcEosFruCode fcEosFruPosition fcEosSysSwitchName fcEosSysSwitchId
Description	An fcEosFruActive trap is generated when a FRU status changes to an active status.
<hr/>	
fcEosFruBackup	
Type Number	7
Product Mapping	Generated when a FRU status changes to a backup status.
Trap Variables	fcEosFruCode fcEosFruPosition fcEosSysSwitchName fcEosSysSwitchId
Description	An fcEosFruBackup trap is generated when a FRU status changes to a backup status.
<hr/>	
fcEosFruUpdate	
Type Number	8
Product Mapping	Generated when a FRU status changes to update/busy.
Trap Variables	fcEosFruCode fcEosFruPosition fcEosSysSwitchName

	fcEosSysSwitchId
Description	An fcEosFruFailed trap is generated when a FRU status changes to update/busy.

fcEosFruFailed

Type Number	9
Product Mapping	Generated when a FRU status changes to a failed status.
Trap Variables	fcEosFruCode fcEosFruPosition fcEosSysSwitchName fcEosSysSwitchId
Description	An fcEosFruFailed trap is generated when a FRU status changes to a failed status.

fcEosLinkBit ErrorEvent

Type Number	10
Product Mapping	Generated when the bit error rate for a link exceeds the threshold.
Trap Variables	fcEosPort Index fcEosPortName fcEosPort WWN fcEosSysSwitchName
Description	An fcEosLinkBit trap is generated when the bit error rate for a link exceeds an allowed threshold.

fcEosLinkNoSignalEvent

Type Number	11
Product Mapping	Generated when there is a loss of signal or sync.
Trap Variables	fcEosPortIndex fcEosPortName fcEosPortWWN fcEosSysSwitchName

Description	An fcEosLinkNoSignalEvent trap is generated when there is a loss of signal or sync.
-------------	---

fcEosLinkNOSEvent

Type Number	12
Product Mapping	Generated when a not operational primitive sequence is received.
Trap Variables	fcEosPortIndex fcEosPortName fcEosPortWWN fcEosSysSwitchName

Description	An fcEosLinkNOSEvent trap is generated when a not operational primitive sequence is received.
-------------	---

fcEosLinkFailureEvent

Type Number	13
Product Mapping	Generated when a primitive sequence timeout occurs.
Trap Variables	fcEosPortIndex fcEosPortName fcEosPortWWN fcEosSysSwitchName

Description	An fcEosLinkFailureEvent trap is generated when a primitive sequence timeout occurs.
-------------	--

fcEosLinkInvalidEvent

Type Number	14
Product Mapping	Generated when an invalid primitive sequence is detected.
Trap Variables	fcEosPortIndex fcEosPortName fcEosPortWWN fcEosSysSwitchName

Description	An fcEosLinkInvalidEvent trap is generated when an invalid primitive sequence is detected.
-------------	--

fcEosLinkAddedEvent

Type Number	15
-------------	----

Product Mapping	Generated when the firmware detects that a new connection has been established on a port.
-----------------	---

Trap Variables	fcEosPortIndex fcEosPortName fcEosPortWWN fcEosSysSwitchName
----------------	---

Description	An fcEosLinkAddedEvent trap is generated when the firmware detects that a new connection has been established on a port.
-------------	--

EXAMPLE: Interpretation of trap information from HP OpenView

The output from HP OpenView for a series of traps is shown below:

```
- Minor Thu May 02 09:29:30 10.235.4.111      NO TRAPD.CONF
FMT FOR .1.3.6.1.2.1.8888.0.1 ARGS(2): [1]
mgmt.mib-2.fcMgmtMIB.fcMgmtObjects.fcMgmtConfig.fcConnUnit
Table.fcConnUnitEntry.fcConnUnitStatus.3.2.0.0.0.0.0.0.0.0.0.0
(Integer): ok [2]
mgmt.mib-2.fcMgmtMIB.fcMgmtObjects.fcMgmtConfig.fcConnUnit
Table.fcConnUnitEntry.fcConnUnitState.3.2.0.0.0.0.0.0.0.0.0.0
(Integer): online

- Minor Thu May 02 09:29:31 10.235.4.111      NO TRAPD.CONF
FMT FOR .1.3.6.1.2.1.8888.0.1 ARGS(2): [1]
mgmt.mib-2.fcMgmtMIB.fcMgmtObjects.fcMgmtConfig.fcConnUnit
Table.fcConnUnitEntry.fcConnUnitStatus.7.0.0.0.0.0.0.0.0.0.0.0
(Integer): ok [2]
mgmt.mib-2.fcMgmtMIB.fcMgmtObjects.fcMgmtConfig.fcConnUnit
Table.fcConnUnitEntry.fcConnUnitState.7.0.0.0.0.0.0.0.0.0.0.0
(Integer): online

- Minor Thu May 02 09:29:46 10.235.4.111      NO TRAPD.CONF
FMT FOR .1.3.6.1.2.1.8888.0.1 ARGS(2): [1]
mgmt.mib-2.fcMgmtMIB.fcMgmtObjects.fcMgmtConfig.fcConnUnit
Table.fcConnUnitEntry.fcConnUnitStatus.3.2.0.0.0.0.0.0.0.0.0.0
(Integer): ok [2]
```

```
mgmt.mib-2.fcMgmtMIB.fcMgmtObjects.fcMgmtConfig.fcConnUnit
Table.fcConnUnitEntry.fcConnUnitState.3.2.0.0.0.0.0.0.0.0.0.0.0
(Integer): online
```

```
- Minor Thu May 02 09:29:47 10.235.4.111 NO TRAPD.CONF
FMT FOR 1.3.6.1.2.1.8888.0.1 ARGS(2): [1]
mgmt.mib-2.fcMgmtMIB.fcMgmtObjects.fcMgmtConfig.fcConnUnit
Table.fcConnUnitEntry.fcConnUnitStatus.7.0.0.0.0.0.0.0.0.0.0.0.0
(Integer): ok [2]
mgmt.mib-2.fcMgmtMIB.fcMgmtObjects.fcMgmtConfig.fcConnUnit
Table.fcConnUnitEntry.fcConnUnitState.7.0.0.0.0.0.0.0.0.0.0.0.0
(Integer): online
```

This output from HP OpenView contains information for 4 traps. Blank lines have been added for clarity. The first step is to determine which trap caused this output. Looking after the words “NO TRAPD.CONF FMT FOR” you can see the numbers 1.3.6.1.2.1.8888.0.1 which identifies this as a switch SCN trap (from table in section 2.3). After the trap OID, the variable bindings are listed. HP OpenView calls them “ARGS” and shows how many have been found in this particular trap (in this case, 2).

The first arg is identified by it’s OID in symbolic form:
 mgmt.mib-2.fcMgmtMIB.fcMgmtObjects.fcMgmtConfig.fcConnUnit
 Table.fcConnUnitEntry.fcConnUnitStatus.

The numbers following fcConnUnitStatus are the unit-id which identifies a particular switch in a fabric. (The unit-id is the first index for all tables in the Fibre Alliance MIB). In this case, these traps are most likely from the EFC Server, which uses a different numbering scheme for the unit-id than the E/OS firmware (see below). In both cases the unit-id is a string of 16 numbers. Following the unit-id is the actual value of the first variable: ok. The value transmitted in the trap is numeric (an integer) but HP OpenView has cross-referenced this numeric value with the MIB definitions to provide the symbolic form (ok). The second variable binding is fcConnUnitState and has the same indexing scheme for unit-id.

Numbering scheme for unit-id (fcConnUnitId) for E/OS and EFCM:

E/OS: WWN(8 numbers).0.0.0.0.0.0.0.0

EFCM: product-code.product-id.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0

In both cases the total length is 16 numbers.

This identifier is used as the first index in all FA MIB tables.

4. 6: Specific trap #3 trap(v1) received from: 172.16.7.243 at 09/25/2002 3:06:45 PM
5. 7: Specific trap #3 trap(v1) received from: 172.16.7.243 at 09/25/2002 3:06:45 PM

As displayed by the MG-SOFT browser, the output above is shown in hierarchical tree form. Trap number 3 has been expanded to show the details of the information contained in the trap. The agent address is the IP address of the switch, and the management address is the address of the PC which was running MG-SOFT. In this case the trap can be identified by the Enterprise (fcMgmtMIB – also known as the FA MIB) and the specific trap number (3), which identifies this as an FA MIB event trap. Lines labeled 4-7 are each for different traps. Referring to trap 3 again, the browser clearly displays the 4 variable bindings contained within an FA MIB event trap. Each variable binding is displayed in the format: OID data-type value.

MIB Definitions: MIB-II

There are eleven groups of objects specified in MIB-II. The E / OS SNMP agent supports eight groups:

- *System Group* . This group provides general information about the managed system.
- *Interfaces Group*
- *Address Translation Group*
- This group is implemented, but the corresponding table may be empty.
- *IP Group*
- *ICMP Group*
- *TCP Group*
- *UDP Group*
- *SNMP Group* This group keeps statistics on the SNMP agent implementation itself.

System Group

sysDescr		
Type	DisplayString(0..255)	
Access	R	
Description	A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating system, and networking software. It is mandatory that this only contain printable ASCII characters.	

sysObjectID		
Type	Object Identifier	
Access	R	

Description The vendor's authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining 'what kind of box' is being managed. For example, if vendor 'Flintstones, Inc.' was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its 'Fred Router'.

sysUpTime

Type TimeTicks

Access R

Description The time (in hundredths of a second) since the network management portion of the system was last re-initialized.

sysContact

Type DisplayString (0..255)

Access R

Description The textual identification of the contact person for this managed node, together with information on how to contact this person.

sysName

Type DisplayString (0..255)

Access RW

Description An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name.

sysLocation

Type DisplayString (0..255)

Access RW

Description The physical location of this node (e.g., 'telephone closet, 3rd floor').

sysServices

Type	INTEGER
Access	R
Description	<p>A value which indicates the set of services that this entity primarily offers. The value is a sum. This sum initially takes the value zero, then, for each layer, L, in the range 1 through 7, that this node performs transactions for, 2 raised to (L - 1) is added to the sum. For example, a node which performs primarily routing functions would have a value of 4 ($2^{(3-1)}$). In contrast, a node which is a host offering application services would have a value of 72 ($2^{(4-1)} + 2^{(7-1)}$). Note that in the context of the Internet suite of protocols, values should be calculated accordingly:</p> <p>layer functionality</p> <ul style="list-style-type: none">1 physical (e.g., repeaters)2 datalink/subnetwork (e.g., bridges)3 internet (e.g., IP gateways)4 end-to-end (e.g., IP hosts)7 applications (e.g., mail relays) <p>For systems including OSI protocols, layers 5 and 6 may also be counted.</p>

Interfaces Group

ifNumber

Type	INTEGER
Access	R
Description	The number of network interfaces (regardless of their current state) present on this system.

Interfaces Table

The interfaces table contains information on the entity's interfaces. Each interface is thought of as being attached to a "subnetwork". Note that this term should not be confused with "subnet" which refers to an addressing partitioning scheme used in the Internet suite of protocols.

ifIndex

Type	INTEGER
Access	R
Description	A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization.

ifDescr

Type	DisplayString(0..255)
Access	R
Description	A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the hardware interface.

ifType

Type	INTEGER
Access	R
Description	<p>The type of interface, distinguished according to the physical/link protocol(s) immediately below the network layer in the protocol stack.</p> <p>Values:</p> <p>other(1), none of the following</p> <p>regular1822(2),</p> <p>hdh1822(3),</p> <p>ddn-x25(4),</p> <p>rfc877-x25(5),</p>

ethernet-csmacd(6),
iso88023-csmacd(7),
iso88024-tokenBus(8),
iso88025-tokenRing(9),
iso88026-man(10),
starLan(11),
proteon-10Mbit(12),
proteon-80Mbit(13),
hyperchannel(14),
fddi(15),
lapb(16),
sdlc(17),
ds1(18), T-1
e1(19), european equivalent of T-1
basicISDN(20),
primaryISDN(21), proprietary serial
propPointToPointSerial(22),
ppp(23),
softwareLoopback(24),
eon(25) CLNP over IP [11]
ethernet-3Mbit(26),
nsip(27), --XNS over IP
slip(28), -- generic SLIP
ultra(29), --ULTRA technologies
ds3(30), --T-3
sip(31), -- SMDS
frame-relay(32)

ifMtu

Type	INTEGER
Access	R
Description	The size of the largest datagram which can be sent/received on the interface, specified in octets. For interfaces that are used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface.

ifSpeed

Type	Gauge
Access	R
Description	An estimate of the interface's current bandwidth in bits per second. For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth.

ifPhysAddress

Type	PhysAddress
Access	R
Description	The interface's address at the protocol layer immediately 'below' the network layer in the protocol stack. For interfaces which do not have such an address (e.g., a serial line), this object should contain an octet string of zero length.

ifAdminStatus

Type	INTEGER
Access	RW
Description	The desired state of the interface. The testing(3) state indicates that no operational packets can be passed.

ifOperStatus

Type	INTEGER
Access	R
Description	The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed.

ifLastChange

Type	TimeTicks
Access	R
Description	The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered prior to the last re-initialization of the local network management subsystem, then this object contains a zero value.

ifInOctets

Type	Counter
Access	R
Description	The total number of octets received on the interface, including framing characters.

ifInUcastPkts

Type	Counter
Access	R
Description	The number of subnetwork-unicast packets delivered to a higher-layer protocol.

ifInNUcastPkts

Type	Counter
Access	R
Description	The number of non-unicast (i.e., subnetwork-broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol.

ifInDiscards

Type	Counter
Access	R
Description	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.

ifInErrors

Type	Counter
Access	R
Description	The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.

ifInUnknownProtos

Type	Counter
Access	R
Description	The number of packets received via the interface which were discarded because of an unknown or unsupported protocol

ifOutOctets

Type	Counter
Access	R
Description	The total number of octets transmitted out of the interface, including framing characters.

ifOutUcastPkts

Type	Counter
Access	R
Description	The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent.

ifOutNUcastPkts

Type	Counter
Access	R
Description	The total number of packets that higher-level protocols requested be transmitted to a non-unicast (i.e., a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent.

ifOutDiscards

Type	Counter
Access	R
Description	The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.

ifOutErrors

Type	Counter
Access	R
Description	The number of outbound packets that could not be transmitted because of errors.

ifOutQLen

Type	Gauge
Access	R
Description	The length of the output packet queue (in packets).

ifSpecific

Type	OBJECT IDENTIFIER
Access	R
Description	A reference to MIB definitions specific to the particular media being used to realize the interface. For example, if the interface is realized

by an ethernet, then the value of this object refers to a document defining objects specific to ethernet. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier, and any conforming implementation of ASN.1 and BER must be able to generate and recognize this value.

Address Translation Group

Implementation of the Address Translation group is mandatory for all systems. Note however that this group is deprecated by MIB-II. That is, it is being included solely for compatibility with MIB-I nodes, and will most likely be excluded from MIB-III nodes. From MIB-II and onwards, each network protocol group contains its own address translation tables.

The Address Translation group contains one table which is the union across all interfaces of the translation tables for converting a NetworkAddress (e.g., an IP address) into a subnetwork-specific address. For lack of a better term, this document refers to such a subnetwork-specific address as a `physical' address. Examples of such translation tables are: for broadcast media where ARP is in use, the translation table is equivalent to the ARP cache; or, on an X.25 network where non-algorithmic translation to X.121 addresses is required, the translation table contains the NetworkAddress to X.121 address equivalences.

atIfIndex

Type	INTEGER
Access	RW
Description	The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

atPhysAddress

Type	PhysAddress
Access	RW
Description	The media-dependent `physical' address. Setting this object to a null string (one of zero length) has the effect of invaliding the

corresponding entry in the atTable object. That is, it effectively disassociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant atPhysAddress object.

atNetAddress

Type	NetworkAddress
Access	RW
Description	The NetworkAddress (e.g., the IP address) corresponding to the media-dependent 'physical' address.

IP Group

ipForwarding

Type	INTEGER
Access	RW
Description	The indication of whether this entity is acting as an IP gateway in respect to the forwarding of datagrams received by, but not addressed to, this entity. IP gateways forward datagrams. IP hosts do not (except those source-routed via the host). Note that for some managed nodes, this object may take on only a subset of the values possible. Accordingly, it is appropriate for an agent to return a 'badValue' response if a management station attempts to change this object to an inappropriate value.

ipDefaultTTL

Type	INTEGER
Access	RW
Description	The default value inserted into the Time-To-Live field of the IP header of datagrams originated at this entity, whenever a TTL value is not supplied by the transport layer protocol.

ipInReceives

Type	Counter
Access	R
Description	The total number of input datagrams received from interfaces, including those received in error.

ipInHdrErrors

Type	Counter
Access	R
Description	The number of input datagrams discarded due to errors in their IP headers, including bad checksums, version number mismatch, other format errors, Time-To-Live exceeded, errors discovered in processing their IP options, etc.

ipInAddrErrors

Type	Counter
Access	R
Description	The number of input datagrams discarded because the IP address in their IP header's destination field was not a valid address to be received at this entity. This count includes invalid addresses (e.g., 0.0.0.0) and addresses of unsupported classes (e.g., Class E). For entities which are not IP Gateways and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address.

ipForwDatagrams

Type	Counter
Access	R
Description	The number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to find a route to forward them to that final destination. In entities which do not act as IP Gateways, this counter will include only those packets which were Source-Routed via this entity, and the Source-Route option processing was successful.

ipInUnknownProtos

Type	Counter
Access	R
Description	The number of locally-addressed datagrams received successfully but discarded because of an unknown or unsupported protocol.

ipInDiscards

Type	Counter
Access	R
Description	The number of input IP datagrams for which no problems were encountered to prevent their continued processing, but which were discarded (e.g., for lack of buffer space). Note that this counter does not include any datagrams discarded while awaiting re-assembly.

ipInDelivers

Type	Counter
Access	R
Description	The total number of input datagrams successfully delivered to IP user-protocols (including ICMP).

ipOutRequests

Type	Counter
Access	R
Description	The total number of IP datagrams which local IP user-protocols (including ICMP) supplied to IP in requests for transmission. Note that this counter does not include any datagrams counted in <code>ipForwDatagrams</code> .

ipOutDiscards

Type	Counter
Access	R

Description The number of output IP datagrams for which no problem was encountered to prevent their transmission to their destination, but which were discarded (e.g., for lack of buffer space). Note that this counter would include datagrams counted in `ipForwDatagrams` if any such packets met this (discretionary) discard criterion.

ipOutNoRoutes

Type Counter

Access R

Description The number of IP datagrams discarded because no route could be found to transmit them to their destination. Note that this counter includes any packets counted in `ipForwDatagrams` which meet this 'no-route' criterion. Note that this includes any datagrams which a host cannot route because all of its default gateways are down.

ipReasmTimeout

Type INTEGER

Access R

Description The maximum number of seconds which received fragments are held while they are awaiting reassembly at this entity.

ipReasmReqds

Type Counter

Access R

Description The number of IP fragments received which needed to be reassembled at this entity.

ipReasmOKs

Type Counter

Access R

Description The number of IP datagrams successfully.

ipReasmFails

Type	Counter
Access	R
Description	The number of failures detected by the IP re-assembly algorithm (for whatever reason: timed out, errors, etc). Note that this is not necessarily a count of discarded IP fragments since some algorithms (notably the algorithm in RFC 815) can lose track of the number of fragments by combining them as they are received.

ipFragOKs

Type	Counter
Access	R
Description	The number of IP datagrams that have been successfully fragmented at this entity.

ipFragFails

Type	Counter
Access	R
Description	The number of IP datagrams that have been discarded because they needed to be fragmented at this entity but could not be, e.g., because their Don't Fragment flag was set.

ipFragCreates

Type	Counter
Access	R
Description	The number of IP datagram fragments that have been generated as a result of fragmentation at this entity.

IP Address Table		The IP address table contains this entity's IP addressing information.
<hr/>		
ipAdEntAddr		
Type	IpAddress	
Access	R	
Description	The IP address to which this entry's addressing information pertains.	
<hr/>		
ipAdEntIfIndex		
Type	INTEGER	
Access	R	
Description	The index value which uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.	
<hr/>		
ipAdEntNetMask		
Type	IpAddress	
Access	R	
Description	The subnet mask associated with the IP address of this entry. The value of the mask is an IP address with all the network bits set to 1 and all the hosts bits set to 0.	
<hr/>		
ipAdEntBcastAddr		
Type	INTEGER	
Access	R	
Description	The value of the least-significant bit in the IP broadcast address used for sending datagrams on the (logical) interface associated with the IP address of this entry. For example, when the Internet standard all-ones broadcast address is used, the value will be 1. This value applies to both the subnet and network broadcasts addresses used by the entity on this (logical) interface.	

ipAdEntReasmMaxSize

Type	INTEGER (0..65535)
Access	R
Description	The size of the largest IP datagram which this entity can re-assemble from incoming IP fragmented datagrams received on this interface.

IP Routing group

The IP routing group contains an entry for each route presently known to this entity.

ipRouteDest

Type	IpAddress
Access	RW
Description	The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. Multiple routes to a single destination can appear in the table, but access to such multiple entries is dependent on the table-access mechanisms defined by the network management protocol in use.

ipRouteIfIndex

Type	INTEGER
Access	RW
Description	The index value which uniquely identifies the local interface through which the next hop of this route should be reached. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

ipRouteMetric1

Type	INTEGER
Access	RW
Description	The primary routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's

ipRouteProto value. If this metric is not used, its value should be set to -1.

ipRouteMetric2

Type	INTEGER
Access	RW
Description	An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

ipRouteMetric3

Type	INTEGER
Access	RW
Description	An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

ipRouteMetric4

Type	INTEGER
Access	RW
Description	An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

ipRouteNextHop

Type	IpAddress
Access	RW
Description	The IP address of the next hop of this route. (In the case of a route bound to an interface which is realized via a broadcast media, the value of this field is the agent's IP address on that interface.)

ipRouteType

Type	INTEGER
Access	RW
Description	<p>The type of route. Note that the values direct(3) and indirect(4) refer to the notion of direct and indirect routing in the IP architecture.</p> <p>Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipRouteTable object. That is, it effectively disassociates the destination identified with said entry from the route identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipRouteType object.</p> <p>Values:</p> <p>other(1), none of the following</p> <p>invalid(2), an invalidated route</p> <p>direct(3), route to directly connected (sub-)network</p> <p>indirect(4) route to a non-localhost/network/sub-network</p>

ipRouteProto

Type	INTEGER
Access	R
Description	<p>The routing mechanism via which this route was learned. Inclusion of values for gateway routing protocols is not intended to imply that hosts should support those protocols.</p> <p>other(1) none of the following</p> <p>local(2) -protocol information, e.g., manually configured entries</p> <p>netmgmt(3) set via a network management protocol</p> <p>icmp(4) e.g., obtained via ICMP, Redirect</p> <p>The remaining values are all gateway routing protocols:</p> <p>egp(5),</p> <p>ggp(6),</p>

hello(7),
rip(8),
is-is(9),
es-is(10),
ciscoIgrp(11),
bbnSpflgp(12),
ospf(13),
bgp(14)

ipRouteAge

Type	INTEGER
Access	RW
Description	The number of seconds since this route was last updated or otherwise determined to be correct. Note that no semantics of 'too old' can be implied except through knowledge of the routing protocol by which the route was learned.

ipRouteMask

Type	IpAddress								
Access	RW								
Description	Indicate the mask to be logical-ANDed with the destination address before being compared to the value in the ipRouteDest field. For those systems that do not support arbitrary subnet masks, an agent constructs the value of the ipRouteMask by determining whether the value of the correspondent ipRouteDest field belong to a class-A, B, or C network, and then using one of: <table><tr><td>mask</td><td>network</td></tr><tr><td>255.0.0.0</td><td>class-A</td></tr><tr><td>255.255.0.0</td><td>class-B</td></tr><tr><td>255.255.255.0</td><td>class-C</td></tr></table> If the value of the ipRouteDest is 0.0.0.0 (a default route), then the mask value is also 0.0.0.0. It should be noted that all IP routing subsystems implicitly use this mechanism.	mask	network	255.0.0.0	class-A	255.255.0.0	class-B	255.255.255.0	class-C
mask	network								
255.0.0.0	class-A								
255.255.0.0	class-B								
255.255.255.0	class-C								

ipRouteMetric5

Type	INTEGER
Access	RW
Description	An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

ipRouteInfo

Type	OBJECT IDENTIFIER
Access	R
Description	A reference to MIB definitions specific to the particular routing protocol which is responsible for this route, as determined by the value specified in the route's ipRouteProto value. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntactically valid object identifier, and any conforming implementation of ASN.1 and BER must be able to generate and recognize this value.

IP Address Translation Table

The IP address translation table contain the IpAddress to physical address equivalences. Some interfaces do not use translation tables for determining address equivalences (e.g., DDN-X.25 has an algorithmic method); if all interfaces are of this type, then the Address Translation table is empty, i.e., has zero entries.

ipNetToMediaIfIndex

Type	INTEGER
Access	RW
Description	The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

ipNetToMediaPhysAddress

Type	PhysAddress
Access	RW

Description	The media-dependent 'physical' address.
-------------	---

ipNetToMediaNetAddress

Type	IpAddress
Access	RW
Description	The IpAddress corresponding to the media-dependent 'physical' address

ipNetToMediaType

Type	INTEGER
Access	RW
Description	The type of mapping. Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipNetToMediaTable. That is, it effectively disassociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipNetToMediaType object.

Values:

other(1),	none of the following
invalid(2),	an invalidated mapping
dynamic(3),	
static(4)	

Additional IP objects

ipRoutingDiscards

Type	Counter
Access	R

Description	The number of routing entries which were chosen to be discarded even though they are valid. One possible reason for discarding such an entry could be to free up buffer space for other routing entries.
-------------	--

ICMP Group

icmpInMsgs

Type	Counter
Access	R
Description	The total number of ICMP messages which the entity received. Note that this counter includes all those counted by icmpInErrors.

icmpInErrors

Type	Counter
Access	R
Description	The number of ICMP messages which the entity received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, etc.).

icmpInDestUnreachs

Type	Counter
Access	R
Description	The number of ICMP Destination Unreachable messages received.

icmpInTimeExcds

Type	Counter
Access	R
Description	The number of ICMP Time Exceeded messages received.

icmpInParmProbs

Type	Counter
Access	R

Description	The number of ICMP Parameter Problem messages received.
-------------	---

icmpInSrcQuenchs

Type	Counter
Access	R
Description	The number of ICMP Source Quench messages received.

icmpInRedirects

Type	Counter
Access	R
Description	The number of ICMP Redirect messages received.

icmpInEchos

Type	Counter
Access	R
Description	The number of ICMP Echo (request) messages received.

icmpInEchoReps

Type	Counter
Access	R
Description	The number of ICMP Echo Reply messages received.

icmpInTimestamps

Type	Counter
Access	R
Description	The number of ICMP Timestamp (request) messages received.

icmpInTimestampReps

Type	Counter
Access	R

Description	The number of ICMP Timestamp Reply messages received.
-------------	---

icmpInAddrMasks

Type	Counter
Access	R
Description	The number of ICMP Address Mask Request messages received.

icmpInAddrMaskReps

Type	Counter
Access	R
Description	The number of ICMP Address Mask Reply messages received

icmpOutMsgs

Type	Counter
Access	R
Description	The total number of ICMP messages which this entity attempted to send. Note that this counter includes all those counted by icmpOutErrors.

icmpOutErrors

Type	Counter
Access	R
Description	The number of ICMP messages which this entity did not send due to problems discovered within ICMP such as a lack of buffers. This value should not include errors discovered outside the ICMP layer such as the inability of IP to route the resultant datagram. In some implementations there may be no types of error which contribute to this counter's value.

icmpOutDestUnreachs

Type	Counter
------	---------

Access	R
Description	The number of ICMP Destination Unreachable messages sent.

icmpOutTimeExcds

Type	Counter
Access	R
Description	The number of ICMP Time Exceeded messages sent.

icmpOutParmProbs

Type	Counter
Access	R
Description	The number of ICMP Parameter Problem messages sent.

icmpOutSrcQuenches

Type	Counter
Access	R
Description	The number of ICMP Source Quench messages sent.

icmpOutRedirects

Type	Counter
Access	R
Description	The number of ICMP Redirect messages sent. For a host, this object will always be zero, since hosts do not send redirects.

icmpOutEchos

Type	Counter
Access	R
Description	The number of ICMP Echo (request) messages sent.

icmpOutEchoReps

Type	Counter
Access	R
Description	The number of ICMP Echo Reply messages sent.

icmpOutTimestamps

Type	Counter
Access	R
Description	The number of ICMP Timestamp (request) messages sent.

icmpOutTimestampReps

Type	Counter
Access	R
Description	The number of ICMP Timestamp Reply messages sent.

icmpOutAddrMasks

Type	Counter
Access	R
Description	The number of ICMP Address Mask Request messages sent.

icmpOutAddrMaskReps

Type	Counter
Access	R
Description	The number of ICMP Address Mask Reply messages sent.

TCP Group

Note that instances of object types that represent information about a particular TCP connection are transient; they persist only as long as the connection in question.

tcpRtoAlgorithm

Type	INTEGER	
Access	R	
Description	The algorithm used to determine the timeout value used for retransmitting unacknowledged octets.	
	Values:	
	other(1),	none of the following
	constant(2)	a constant rto
	rsre(3)	MIL-STD-1778, Appendix B
	vanj(4)	Van Jacobson's algorithm [10]

tcpRtoMin

Type	INTEGER	
Access	R	
Description	The minimum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the LBOUND quantity described in RFC 793.	

tcpRtoMax

Type	INTEGER	
Access	R	
Description	The maximum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the	

timeout algorithm is rsre(3), an object of this type has the semantics of the UBOUND quantity described in RFC 793.

tcpMaxConn

Type	INTEGER
Access	R
Description	The limit on the total number of TCP connections the entity can support. In entities where the maximum number of connections is dynamic, this object should contain the value -1.

tcpActiveOpens

Type	Counter
Access	R
Description	The number of times TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state.

tcpPassiveOpens

Type	Counter
Access	R
Description	The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state.

tcpAttemptFails

Type	Counter
Access	R
Description	The number of times TCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state.

tcpEstabResets

Type	Counter
------	---------

Access	R
Description	The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state.

tcpCurrEstab

Type	Gauge
Access	R
Description	The number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.

tcpInSegs

Type	Counter
Access	R
Description	The total number of segments received, including those received in error. This count includes segments received on currently established connections.

tcpOutSegs

Type	Counter
Access	R
Description	The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets.

tcpRetransSegs

Type	Counter
Access	R
Description	The total number of segments retransmitted. That is, the number of TCP segments transmitted containing one or more previously transmitted octets.

TCP Connection Table

The TCP connection table contains information about this entity's existing TCP connections.

tcpConnState

Type	INTEGER
Access	RW
Description	The state of this TCP connection. The only value which may be set by a management station is deleteTCB(12). Accordingly, it is appropriate for an agent to return a 'badValue' response if a management station attempts to set this object to any other value. If a management station sets this object to the value deleteTCB(12), then this has the effect of deleting the TCB (as defined in RFC 793) of the corresponding connection on the managed node, resulting in immediate termination of the connection. As an implementation-specific option, an RST segment may be sent from the managed node to the other TCP endpoint (note however that RST segments are not sent reliably).

Values:

closed(1),
listen(2),
synSent(3),
synReceived(4),
established(5),
finWait1(6),
finWait2(7),
closeWait(8),
lastAck(9),
closing(10),
timeWait(11),
deleteTCB(12)

tcpConnLocalAddress

Type	IpAddress
Access	R

Description	The local IP address for this TCP connection. In the case of a connection in the listen state which is willing to accept connections for any IP interface associated with the node, the value 0.0.0.0 is used.
-------------	--

tcpConnLocalPort

Type	INTEGER (0..65535)
Access	R
Description	The local port number for this TCP connection.

tcpConnRemAddress

Type	IpAddress
Access	R
Description	The remote IP address for this TCP connection.

tcpConnRemPort

Type	INTEGER (0..65535)
Access	R
Description	The remote port number for this TCP connection.

Additional TCP Objects

tcpInErrs

Type	Counter
Access	R
Description	The total number of segments received in error (e.g., bad TCP checksums)

tcpOutRsts

Type	Counter
Access	R
Description	The number of TCP segments sent containing the RST flag.

UDP Group

udpInDatagrams

Type	Counter
Access	R
Description	The total number of UDP datagrams delivered to UDP users.

udpNoPorts

Type	Counter
Access	R
Description	The total number of received UDP datagrams for which there was no application at the destination port.

udpInErrors

Type	Counter
Access	R
Description	The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port.

udpOutDatagrams

Type	Counter
Access	R
Description	The total number of UDP datagrams sent from this entity.

UDP Listener Table

The UDP listener table contains information about this entity's UDP end-points on which a local application is currently accepting datagrams.

udpLocalAddress

Type	IpAddress
Access	R
Description	The local IP address for this UDP listener. In the case of a UDP listener which is willing to accept datagrams for any IP interface associated with the node, the value 0.0.0.0 is used.

udpLocalPort

Type	INTEGER (0..65535)
Access	R
Description	The local port number for this UDP listener.

SNMP Group

Some of the objects defined below will be zero-valued in those SNMP implementations that are optimized to support only those functions specific to either a management agent or a management station. In particular, it should be observed that the objects below refer to an SNMP entity, and there may be several SNMP entities residing on a managed node (e.g., if the node is hosting acting as a management station).

snmpInPkts

Type	Counter
Access	R
Description	The total number of messages delivered to the SNMP entity from the transport service.

snmpOutPkts

Type	Counter
Access	R
Description	The total number of SNMP messages which were passed from the SNMP protocol entity to the transport service.

snmpInBadVersions

Type	Counter
Access	R
Description	The total number of SNMP messages which were delivered to the SNMP protocol entity and were for an unsupported SNMP version.

snmpInBadCommunityNames

Type	Counter
Access	R
Description	The total number of SNMP messages delivered to the SNMP protocol entity which used a SNMP community name not known to said entity.

snmpInBadCommunityUses

Type	Counter
Access	R
Description	The total number of SNMP messages delivered to the SNMP protocol entity which represented an SNMP operation which was not allowed by the SNMP community named in the messages.

snmpInASNParseErrs

Type	Counter
Access	R
Description	The total number of ASN.1 or BER errors encountered by the SNMP protocol entity when decoding received SNMP messages.

snmpInTooBigs

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is 'tooBig.'

snmpInNoSuchNames

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is 'noSuchName.'

snmpInBadValues

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is 'badValue.'

snmpInReadOnlys

Type	Counter
Access	R
Description	The total number valid SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is 'readOnly.' It should be noted that it is a protocol error to generate an SNMP PDU which contains the value 'readOnly' in the error-status field, as such this object is provided as a means of detecting incorrect implementations of the SNMP.

snmpInGenErrs

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is 'genErr.'

snmpInTotalReqVars

Type	Counter
Access	R
Description	The total number of MIB objects which have been retrieved successfully by the SNMP protocol entity as the result of receiving valid SNMP Get-Request and Get-Next PDUs.

snmpInTotalSetVars

Type	Counter
Access	R
Description	The total number of MIB objects which have been altered successfully by the SNMP protocol entity as the result of receiving valid SNMP Set-Request PDUs.

snmpInGetRequests

Type	Counter
Access	R
Description	The total number of SNMP Get-Request PDUs which have been accepted and processed by the SNMP protocol entity.

snmpInGetNexts

Type	Counter
Access	R
Description	The total number of SNMP Get-Next PDUs which have been accepted and processed by the SNMP protocol entity.

snmpInSetRequests

Type	Counter
Access	R
Description	The total number of SNMP Set-Request PDUs which have been accepted and processed by the SNMP protocol entity.

snmpInGetResponses

Type	Counter
Access	R
Description	The total number of SNMP Get-Response PDUs which have been accepted and processed by the SNMP protocol entity.

snmpInTraps

Type	Counter
Access	R
Description	The total number of SNMP Trap PDUs which have been accepted and processed by the SNMP protocol entity.

snmpOutTooBigs

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is `tooBig.'

snmpOutNoSuchNames

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status is `noSuchName.'

snmpOutBadValues

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is `badValue.'

snmpOutGenErrs

Type	Counter
Access	R
Description	The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is `genErr.'

snmpOutGetRequests

Type	Counter
Access	R
Description	The total number of SNMP Get-Request PDUs which have been generated by the SNMP protocol entity.

snmpOutGetNexts

Type	Counter
Access	R
Description	The total number of SNMP Get-Next PDUs which have been generated by the SNMP protocol entity.

snmpOutSetRequests

Type	Counter
Access	R
Description	The total number of SNMP Set-Request PDUs which have been generated by the SNMP protocol entity.

snmpOutGetResponses

Type	Counter
Access	R
Description	The total number of SNMP Get-Response PDUs which have been generated by the SNMP protocol entity.

snmpOutTraps

Type	Counter
Access	R
Description	The total number of SNMP Trap PDUs which have been generated by the SNMP protocol entity.

snmpEnableAuthenTraps

Type	INTEGER
Access	RW
Description	<p>Indicates whether the SNMP agent process is permitted to generate authentication-failure traps. The value of this object overrides any configuration information; as such, it provides a means whereby all authentication-failure traps may be disabled. Note that it is strongly recommended that this object be stored in non-volatile memory so that it remains constant between re-initializations of the network management system.</p> <p>Values:</p> <p>enabled(1),</p> <p>disabled(2)</p>

Fabric Element Management MIB

There are five groups of objects defined in the Fabric Element Management MIB.

Fabric Element Management MIB Tables

Predefined types

DisplayString

Syntax OCTET STRING

MilliSeconds

Syntax INTEGER (0..2147383647)
 $2^{31} - 1$

MicroSeconds

Syntax INTEGER (0..2147383647)

FcNameId

Syntax OCTET STRING (SIZE(8))

Description World wide Name or Fibre Channel Name associated with an FC entity. It's a Network_Destination_ID or Network_Source_ID composed of a value up to 60 bits wide, occupying the remaining 8 bytes while the first nibble identifies the format of the Name_Identifier with hex values: 0: ignored, 1: IEEE 48-bit address, 2: IEEE extended, 3: Locally assigned, 4: 32-bit IP address.

FabricName

Syntax FcNameId

Description The Name Identifier of a Fabric. Each Fabric shall provide a unique Fabric Name. Only the following formats are allowed: IEEE48, and Local.

FcPortName

Syntax	FcNameId
Description	The Name Identifier associated with a port. Only the following formats are allowed: IEEE48, IEEE extended, and Local.

FcAddressId

Syntax	OCTET STRING (SIZE (3))
Description	Fibre Channel Address Identifier. A 24-bit value unique within the address space of a Fabric.

FcRxDataFieldSize

Syntax	INTEGER (128..2112)
Description	Receive Data_Field Size.

FcBbCredit

Syntax	INTEGER (0..32767)
Description	Buffer-to-buffer Credit.

FcphVersion

Syntax	INTEGER (0..255)
Description	

FcStackedConnMode

Syntax	INTEGER
Description	The values are defined as follow: none(1), transparent(2), lockedDown(3).

FcCosCap

Syntax	INTEGER (0..127)
Description	bit 0 – Class F bit 1 – Class 1

bit 2 -- Class 2
 bit 3 -- Class 3
 bit 4 -- Class 4
 bit 5 -- Class 5
 bit 6 -- Class 6
 bit 7 -- reserved for future

Fc0BaudRate

Syntax	INTEGER
Description	The values are defined as follow:
other(1)	none of below
oneEighth(2)	155 Mbaud (12.5MB/s)
quarter(4)	266 Mbaud (25.0MB/s)
half(8)	532 Mbaud (50.0MB/s)
full(16)	1 Gbaud (100MB/s)
double(32)	2 Gbaud (200MB/s)
quadruple(64)	4 Gbaud (400MB/s)

Fc0BaudRateCap

Syntax	INTEGER (0..127)
Description	
bit 0	other
bit 1	oneEighth
bit 2	quarter
bit 3	half
bit 4	full
bit 5	double
bit 6	quadruple
bit 7	reserved for future

Fc0MediaCap

Syntax	INTEGER (0..65535)
Description	
bit 0	unknown
bit 1	single mode fibre (sm)
bit 2	multi-mode fibre 50 micron (m5)
bit 3	multi-mode fibre 62.5 micron (m6)
bit 4	video cable (tv)
bit 5	miniature cable (mi)
bit 6	shielded twisted pair (stp)
bit 7	twisted wire (tw)
bit 8	long video (lv)
bits 9-15	reserved for future use.

Fc0Medium

Syntax	INTEGER
Description	The values are defined as follows: unknown(1) sm(2) m5(4) m6(8) tv(16) mi(32) stp(64) tw(128) lv(256)

Fc0TxType

Syntax	INTEGER
Description	The values are defined as follows:

unknown(1)
 longWaveLaser(2) – (LL)
 shortWaveLaser(3)—(SL)
 longWaveLED(4) – (LE)
 electrical(5) – (EL)
 shortWaveLaser-noOFC(6) – (SN)

Fc0Distance

Syntax	INTEGER
Description	The values are defined as follow: unknown(1), long(2), intermediate(3), short(4).

FcFeModuleCapacity

Syntax	INTEGER (1..256)
Description	

FcFeFxPortCapacity

Syntax	INTEGER (1..256)
Description	

FcFeModuleIndex

Syntax	INTEGER (1..256)
Description	

FcFeFxPortIndex

Syntax	INTEGER (1..256)
--------	------------------

FcFeNxPortIndex

Syntax	INTEGER (1..126)
--------	------------------

FcFxPortMode

Syntax	INTEGER
Description	The values are defined as follow: unknown(1), fPort(2), flPort(3).

FcBbCreditModel

Syntax	INTEGER
Description	The values are defined as follow: regular(1), alternate(2).

MIB objects defined in the Fabric Element MIB:

fcFabricName

Type	FabricName
Access	R
Description	The Name_Identifier of the Fabric to which this Fabric Element belongs.

FcElementName

Type	FcNameId
Access	R
Description	The Name_Identifier of the Fabric Element.

FcFeModuleCapacity

Type	FcFeModuleCapacity
Access	R
Description	The maximum number of modules in the Fabric Element, regardless of their current state.

Module Table

A table that contains one entry for each module in the Fabric Element, containing information about the modules

Fabric Element MIB Object Name

fcFeModuleDescr

Type	DisplayString (SIZE(256))
Provided By	McK DEV_TBL
Access	R
Description	<p>A textual description of the module. This value should include the full name and version identification of the module. It should contain printable ASCII characters.</p> <p>This string should be derived from VPD information stored in the FRU EEPROM.</p>

FcFeModuleObjectID

Type	OBJECT IDENTIFIER
Provided By	SNMP
Access	R
Description	This is a fixed object identifier assigned from the McDATA enterprise subtree (1.3.6.1.4.1.289.2.1.1.2).

fcFeModuleOperStatus

Type	INTEGER
Provided By	SNMP
Access	R
Description	<p>This object indicates the operational status of the module: online(1) – the module is functioning properly; offline(2) – the module is not available; testing(3) – the module is under testing; and faulty(4) – the module is defective in some way.</p> <p>The status is evaluated from <code>fcFPortPhysOperStatus</code> as following order.</p> <p>Testing(3): the module is under testing if all four ports on the current module are testing.</p> <p>faulty(4): the module is defective if any of the ports on the current module is faulty.</p> <p>Online(1): the module is functioning properly if any of the ports on the current module is online or testing.</p>

offline(2): the module is not available if any of the ports on the current module is offline.

FcFeModuleLastChange

Type	TIMETICKS
Provided By	SNMP
Access	R
Description	This object contains the value of sysUpTime when the module entered its current operational status. A value of zero indicates that the operational status of the module has not changed since the agent last restarted. This is SS_TIM_RD_TICKS(MILLISEC) * 10.

fcFeModuleFxPortCapacity

Type	FcFeFxPortCapacity
Provided By	AS
Access	R
Description	The number of Fx_Port that can be contained within the module. Within each module, the ports are uniquely numbered in the range from 1 to fcFeModuleFx_PortCapacity inclusive. However, the numbers are not required to be contiguous. This is AS_glob.prod_cnfg_ptr->ports_per_module.

fcFeModuleName

Type	FcNameId
Provided By	PCP
Access	R
Description	The Name_Identifier of the module. This is the port module world wide name.

Fx_Port Configuration Table

A table that contains one entry for each Fx_Port in the Fabric Element, containing configuration and service parameters of the Fx_Ports.

fcFxConfFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	<p>This object identifies the Fx_Port within the module. This number ranges from 1 to the value of <code>fcFeModulePortCapacity</code> for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.</p> <p>This number ranges from 1 to AS_glob.prod_cnfg_ptr->ports_per_module.</p>

FcFxPortName

Type	FcPortName
Provided By	PCP
Access	R
Description	<p>The name identifier of this Fx_Port. Each Fx_Port has a unique port name within the address space of the Fabric.</p> <p>This is the WWN assigned to the port.</p>

FcFxPortFcphVersionHigh

Type	FcphVersion
Provided By	FC2
Access	R
Description	<p>The highest or most recent version of FC-PH that the Fx_Port is configured to support. Since the switch is not capable of changing its support for FC-PH version, the version reported is the one currently in use for this port. If there is no device logged in, then the value is 0.</p> <p>If a device is logged in, the values reported are:</p> <p>6 = FC-PH 4.0</p> <p>7 = FC-PH 4.1</p> <p>8 = FC-PH 4.2</p> <p>9 = FC-PH 4.3</p>

0x10 = FC-PH2

0x20 = FC-PH3

FcFxPortFcphVersionLow

Type	FcphVersion
Provided By	FC2
Access	R
Description	The lowest or earliest version of FC-PH that the Fx_Port is configured to support. Since the switch is not capable of changing its support for FC-PH version, the version reported is the one currently in use for this port. If there is no device logged in, then the value is 0. For values see FcFxPortFcphVersionHigh on page 2-69.

FcFxPortBbCredit

Type	FcBbCredit
Provided By	PCP
Access	R
Description	The total number of receive buffers available for holding Class 1 connect-request, Class 2 or 3 frames from the attached Nx_Port. It is for buffer-to-buffer flow control in the direction from the attached Nx_Port (if applicable) to F_port.

FcFxPortRxBufSize

Type	FcRxDataFieldSize
Provided By	LOGIN SERVER
Access	R
Description	The largest Data_Field Size (in octets) for an FT_1 frame that can be received by the Fx_Port. This is fixed at 2112.

FcFxPortRatov

Type	Milliseconds
------	--------------

Provided By	PCP
Access	R
Description	The Resource_Allocation_Timeout Value configured for the Fx_Port. This is used as the timeout value for determining when to reuse an NxPort resource such as a Recovery_Qualifier. It represents E_D_TOV (see next object) plus twice the maximum time that a frame may be delayed within the Fabric and still be delivered.

FcFxPortEdtov

Type	Milliseconds
Provided By	PCP
Access	R
Description	The E_D_TOV value configured for the Fx_Port. The Error_Detect_Timeout Value is used as the timeout value for detecting an error condition.

FcFxPortCosSupported

Type	FcCosCap
Provided By	SNMP
Access	R
Description	A value indicating the set of Classes of Service supported by the Fx_Port. This is fixed at CLASS_2 CLASS_3 (0x0C).

fcFxPortIntermixSupported

Type	INTEGER
Provided By	SNMP
Access	R
Description	A flag indicating whether or not the Fx_Port supports an Intermixed Dedicated Connection. The values are defined as follow: yes(1) and no(2). This is fixed at no(2).

FcFxPortStackedConnMode

Type	FcStackedConnMode
Provided By	SNMP
Access	R
Description	A value indicating the mode of Stacked Connect supported by the Fx_Port. This is fixed at none(1).

FcFxPortClass2SeqDeliv

Type	INTEGER
Provided By	SNMP
Access	R
Description	A flag indicating whether or not Class 2 Sequential Delivery is supported by the Fx_Port. The values are defined as follow: yes(1) and no(2). This is fixed at yes(1).

FcFxPortClass3SeqDeliv

Type	INTEGER
Provided By	SNMP
Access	R
Description	A flag indicating whether or not Class 3 Sequential Delivery is supported by the Fx_Port. The values are defined as follow: yes(1) and no(2). This is fixed at yes(1).

FcFxPortHoldTime

Type	MicroSeconds
Provided By	PCP
Access	R

Description The maximum time (in microseconds) that the Fx_Port shall hold a frame before discarding the frame if it is unable to deliver the frame. The value 0 means that the Fx_Port does not support this parameter.

This is equal to quarter of d the E_D_TOV which is obtained from PCP.

FcFxPortBaudRate

Type Fc0BaudRate

Provided By FPM

Access R

Description The FC-0 baud rate of the Fx_Port.

One of these values, or no value will be returned.

0x10, 1 Gbaud (100 MB/s)

0x20, 2 Gbaud (200 MB/s)

0x40 4 Gbaud (400 MB/s)

FcFxPortMedium

Type Fc0Medium

Provided By FPM

Access R

Description The FC-0 medium of the Fx_Port.

The value is a bitwise OR of these values:

0x02 Single Mode fibre

0x04 Multi-mode fibre 50 micron

0x08 Multi-mode fibre 62.5 micron

Or it will be unknown (0x01) if no information is available.

FcFxPortTxType

Type Fc0TxType

Provided By FPM

Access R

Description The FC-0 transmitter type of the Fx_Port.

- 1 Unknown (long distance laser)
- 2 LongwaveLaser (LC version)
- 3 ShortwaveLaser
- 6 ShortwaveLaser-no OFC

FcFxPortDistance

Type Fc0Distance

Provided By FPM

Access R

Description The FC-0 distance range of the Fx_Port transmitter.

- 1 Unknown
- 2 Long
- 3 Intermediate
- 4 Short

Fx_Port Operation Table A table that contains one entry for each Fx_Port in the Fabric Element, operational status and parameters of the Fx_Ports.

fcFxPortOperFxPortIndex

Type FcFeFxPortIndex

Provided By SNMP

Access R

Description This object identifies the Fx_Port within the module. This number ranges from 1 to the value of `fcFeModulePortCapacity` for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortID

Type FcAddressId

Provided By Login Server

Access R

Description	<p>The address identifier by which this Fx_Port is identified within the Fabric. The Fx_Port may assign its address identifier to its attached NxPort(s) during Fabric Login.</p> <p>Return a port id if the port is logged into the fabric, otherwise this address is 000000 in FCEOS.</p>
-------------	---

fcFPortAttachedPortName

Type	FcPortName
Provided By	Login Server
Access	R
Description	<p>The port name of the attached N_Port, if applicable. If the value of this object is '0000000000000000'H, this F_Port has no NxPort attached to it. This variable has been deprecated and may be implemented for backward compatibility. Not supported for NL_ports.</p>

FcFPortConnectedPort

Type	FcAddressId
Provided By	SNMP
Access	R
Description	<p>The address identifier of the destination Fx_Port with which this Fx_Port is currently engaged in a either a Class 1 or loop connection. If the value of this object is '000000'H, this Fx_Port is not engaged in a class 1 connection. This variable has been deprecated and may be implemented for backward compatibility.</p> <p>This address is fixed at 0x000000.</p>

FcFxPortBbCreditAvailable

Type	Gauge
Provided By	PSCC
Access	R
Description	<p>The number of buffers currently available for receiving frames from the attached port in the buffer-to-buffer flow control. The value should be less than or equal to <i>fcFxPortBbCredit</i>.</p>

FcFxPortOperMode

Type	FcFxPortMode
Provided By	AS
Access	R
Description	The current operational mode of the Fx_Port. This value is F_Port(2) if the port_state_data is unavailable or the port is an F_Port, or unknown(1) for the other port state.

FcFxPortAdminMode

Type	FcFxPortMode
Provided By	AS
Access	R
Description	The desired operational mode of the Fx_Port. This value is F_Port(2) if the port_state_data is unavailable or the port is an F_Port, or unknown(1) for the other port state.

Fx_Port Physical Level Table	A table that contains one entry for each Fx_Port in the Fabric Element, containing physical level status and parameters of the Fx_Ports
-------------------------------------	---

fcFxPortPhysFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortPhysAdminStatus

Type	INTEGER
Provided By	PCP, FPM

Access	R/W
Description	<p>The desired state of the Fx_Port. A management station may place the Fx_Port in a desired state by setting this object accordingly. The testing(3) state indicates that no operational frames can be passed. When a Fabric Element initializes, all Fx_Port start with <code>fcFxPortPhysAdminStatus</code> in the offline(2) state.</p> <p>As the result of either explicit management action or per configuration information accessible by the Fabric Element, <code>fcFxPortPhysAdminStatus</code> is then changed to either the online(1) or testing(3) states, or remains in the offline state. The values are defined as follow: online(1) – place port online, offline(2) – take port offline, testing (3).</p> <p>If the port cannot be set to testing because it is inactive or in a failed state, the return value will be <code>resource_unavailable(13)</code>.</p>

FcFxPortPhysOperStatus

Type	INTEGER
Provided By	FPM, SNMP
Access	R
Description	<p>The current operational status of the Fx_Port. The testing(3) status indicates that no operational frames can be passed. If <code>fcFxPortPhysAdminStatus</code> is offline(2) then <code>fcFxPortPhysOperStatus</code> should be offline(2).</p> <p>If <code>fcFxPortPhysAdminStatus</code> is changed to online(1) then <code>fcFxPortPhysOperStatus</code> should change to online(1) if the Fx_Port is ready to accept Fabric Login request from the attached NxPort; it should proceed and remain in the link-failure(4) state if and only if there is a fault that prevents it from going to the online(1) state.</p> <p>The values are defined as online(1) – Login may proceed, offline(2) – Login cannot proceed, testing(3) – port is under test, link-failure(4) – failure after online/ testing.</p>

FcFxPortPhysLastChange

Type	TimeTicks
Provided By	SNMP
Access	R

Description The value of sysUpTime at the time the Fx_Port entered its current operational status. A value of zero indicates that the Fx_Port's operational status has not changed since the agent last restarted.

This is $SS_TIM_RD_TICKS(MILLISEC) * 10$.

FcFxPortPhysRttov

Type MilliSeconds

Provided By SNMP

Access R

Description The Receiver_Transmitter_Timeout value of the Fx_Port. This is used by the receiver logic to detect Loss of Synchronization.

This value is fixed at 100ms.

Fx_Port Fabric Login Table

An entry containing service parameters established from a successful Fabric Login.

fcFxlogiFxPortIndex

Type FcFeFxPortIndex

Provided By SNMP

Access R

Description This object identifies the Fx_Port within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxlogiNxPortIndex

Type FcFeNxPortIndex

Provided By SNMP

Access R

Description The object identifies the associated NxPort in the attachment for which the entry contains information.

FcFxPortFcphVersionAgreed

Type	FcphVersion
Provided By	Login Server
Access	R
Description	The version of FC-PH that the Fx_Port has agreed to support from the Fabric Login.

FcFxPortNxPortBbCredit

Type	FcBbCredit
Provided By	Login Server
Access	R
Description	The total number of buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames to be transmitted to the attached NxPort. It is for buffer- to-buffer flow control in the direction from Fx_Port to Nx_Port. The buffer-to-buffer flow control mechanism is indicated in the respective <code>fcFxPortBbCreditModel</code> .

FcFxPortNxPortRxDataFieldSize

Type	FcRxDataFieldSize
Provided By	Login Server
Access	R
Description	The Receive Data Field Size of the attached NxPort. This is a binary value that specifies the largest Data Field Size for an FT_1 frame that can be received by the NxPort. The value is in number of bytes and ranges from 128 to 2112 inclusive.

FcFxPortCosSuppAgreed

Type	FcCosCap
Provided By	Login Server
Access	R

Description	A variable indicating that the attached NxPort has requested the Fx_Port for the support of classes of services and the Fx_Port has granted the request.
-------------	--

FcFxPortIntermixSuppAgreed

Type	INTEGER
Provided By	SNMP
Access	R
Description	A variable indicating that the attached Nx_Port has requested the Fx_Port for the support of Intermix and the Fx_Port has granted the request. This flag is only valid if Class 1 service is supported. The values are defined as yes(1) and no(2). This is always no(2).

FcFxPortStackedConnModeAgreed

Type	FcStackedConnMode
Provided By	SNMP
Access	R
Description	A variable indicating whether the Fx_Port has agreed to support stacked connect from the Fabric Login. This is only meaningful if Class 1 service has been agreed. This is always none(1).

FcFxPortClass2SeqDelivAgreed

Type	INTEGER
Provided By	Login Server
Access	R
Description	A variable indicating whether the Fx_Port has agreed to support Class 2 sequential delivery from the Fabric Login. This is only meaningful if Class 2 service has been agreed. The values are defined as yes(1) and no(2).

FcFxPortClass3SeqDelivAgreed

Type	INTEGER
Provided By	Login Server
Access	R
Description	A flag indicating whether the Fx_Port has agreed to support Class 3 sequential delivery from the Fabric Login. This is only meaningful if Class 3 service has been agreed. The values are defined as yes(1) and no(2).

FcFxPortNxPortName

Type	FcPortName
Provided By	Login Server
Access	R
Description	<p>The port name of the attached Nx_Port, if applicable. If the value of this object is '0000000000000000'H, this Fx_Port has no Nx_Port attached to it.</p> <p>This is the world wide Name of the attached Nx_Port. It's same as fcFPortAttachedPortName.</p>

FcFxPortConnectedNxPort

Type	FcAddressId
Provided By	SNMP
Access	R
Description	<p>The address identifier of the destination Fx_Port with which this Fx_Port is currently engaged in a either a Class 1 or loop connection. If the value of this object is '000000'H, this Fx_Port is not engaged in a connection.</p> <p>This is fixed at '000000'H.</p>

fcFxPortBbCreditModel

Type	FcBbCreditModel
Provided By	SNMP

Access	R
Description	This object identifies the BB_Credit model used by the Fx_Port. The regular model refers to the Buffer-to-Buffer flow control mechanism defined in FC-PH [1] is used between the F_Port and the N_Port. For FL_Ports, the Alternate Buffer-to-Buffer flow control mechanism as defined in FC-AL [4] is used between the FL_Port and any attached NL_Ports. This is fixed at regular(1).

Fx_Port Error Table A table that contains one entry for each Fx_Port, counters that record the numbers of errors detected.

fcFxPortErrorFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of <code>fcFeModulePortCapacity</code> for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortLinkFailures

Type	Counter
Provided By	PSCC
Access	R
Description	The number of link failures detected by this Fx_Port.

FcFxPortSyncLosses

Type	Counter
Provided By	PSCC
Access	R
Description	The number of loss of synchronization detected by the Fx_Port.

FcFxPortSigLosses

Type	Counter
Provided By	PSCC
Access	R
Description	The number of loss of signal detected by the Fx_Port.

FcFxPortPrimSeqProtoErrors

Type	Counter
Provided By	PSCC
Access	R
Description	The number of primitive sequence protocol errors detected by the Fx_Port.

FcFxPortInvalidTxWords

Type	Counter
Provided By	PSCC
Access	R
Description	The number of invalid transmission word detected by the Fx_Port.

FcFxPortInvalidCrcs

Type	Counter
Provided By	PSCC
Access	R
Description	The number of invalid CRC detected by the Fx_Port.

FcFxPortDelimiterErrors

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Delimiter Errors detected by this Fx_Port.

FcFxPortAddressIdErrors

Type	Counter
Provided By	PSCC
Access	R
Description	The number of address identifier errors detected by this Fx_Port.

FcFxPortLinkResetIns

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Link Reset Protocol received by this Fx_Port from the attached Nx_Port.

FcFxPortLinkResetOuts

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Link Reset Protocol issued by this Fx_Port to the attached Nx_Port.

FcFxPortOlsIns

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Offline Sequence received by this Fx_Port.

FcFxPortOlsOuts

Type	Counter
Provided By	PSCC
Access	R

Description The number of Offline Sequence issued by this Fx_Port.

Class 1 Accounting table A table that contains one entry for each Fx_Port in the Fabric Element, Class 1 accounting information. These entries are all zero except for the index, since class 1 is not supported.

fcFxPortC1AcctFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of <code>fcFeModulePortCapacity</code> for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.

FcFxPortC1InConnections

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 1 connections successfully established in which the attached Nx_Port is the source of the connect-request. This value is fixed at 0.

FcFxPortC1OutConnections

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 1 connections successfully established in which the attached Nx_Port is the destination of the connect-request. This value is fixed at 0.

FcFxPortC1FbsyFrames

Type	Counter
------	---------

Provided By	SNMP
Access	R
Description	The number of F_BSY frames generated by this Fx_Port against Class 1 connect-request. This value is fixed at 0.

FcFxPortC1FrjtFrames

Type	Counter
Provided By	SNMP
Access	R
Description	The number of F_RJT frames generated by this Fx_Port against Class 1 connect-request. This value is fixed at 0.

FcFxPortC1ConnTime

Type	Counter
Provided By	SNMP
Access	R
Description	The cumulative time that this Fx_Port has been engaged in Class 1 connection. The amount of time of each connection is counted in octets from after a connect- request has been accepted until the connection is disengaged, either by an EOFdt or Link Reset. This value is fixed at 0.

FcFxPortC1InFrames

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 1 frames (other than Class 1 connect-request) received by this Fx_Port from its attached Nx_Port. This value is fixed at 0.

FcFxPortC1OutFrames

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 1 frames (other than Class 1 connect-request) delivered through this Fx_Port to its attached Nx_Port. This value is fixed at 0.

FcFxPortC1InOctets

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 1 frame octets, including the frame delimiters, received by this Fx_Port from its attached Nx_Port. This value is fixed at 0.

FcFxPortC1OutOctets

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 1 frame octets, including the frame delimiters, delivered through this Fx_Port its attached Nx_Port. This value is fixed at 0.

FcFxPortC1Discards

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 1 frames discarded by this Fx_Port. This value is fixed at 0.

Class 2 Accounting table

A table that contains one entry for each Fx_Port in the Fabric Element, Class 2 accounting information recorded since the management agent has re-initialized.

fcFxPortC2AcctFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of <code>fcFeModulePortCapacity</code> for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortC2InFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 2 frames received by this Fx_Port from its attached Nx_Port.

FcFxPortC2OutFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 2 frames delivered through this Fx_Port to its attached Nx_Port.

FcFxPortC2InOctets

Type	Counter
Provided By	PSCC
Access	R

Description	The number of Class 2 frame octets, including the frame delimiters, received by this Fx_Port from its attached Nx_Port.
-------------	---

FcFxPortC2OutOctets

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 2 frame octets, including the frame delimiters, delivered through this Fx_Port to its attached Nx_Port.

FcFxPortC2Discards

Type	Counter
Provided By	SNMP
Access	R
Description	The number of Class 2 frames discarded by this Fx_Port. This value is not supported. It's always zero.

FcFxPortC2FbsyFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of F_BSY frames generated by this Fx_Port against Class 2 frames.

FcFxPortC2FrjtFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of F_RJT frames generated by this Fx_Port against Class 2 frames.

Class 3 Accounting table

A table that contains one entry for each Fx_Port in the Fabric Element, Class 3 accounting information recorded since the management agent has re-initialized.

fcFxPortC3AcctFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of <code>fcFeModulePortCapacity</code> for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortC3InFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 3 frames received by this Fx_Port from its attached Nx_Port.

FcFxPortC3OutFrames

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 3 frames delivered through this Fx_Port to its attached Nx_Port.

FcFxPortC3InOctets

Type	Counter
Provided By	PSCC
Access	R

Description	The number of Class 3 frame octets, including the frame delimiters, received by this Fx_Port from its attached Nx_Port.
-------------	---

FcFxPortC3OutOctets

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 3 frame octets, including the frame delimiters, delivered through this Fx_Port to its attached Nx_Port.

FcFxPortC3Discards

Type	Counter
Provided By	PSCC
Access	R
Description	The number of Class 3 frames discarded by this Fx_Port.

Fx_Port Capability Table	A table that contains one entry for each Fx_Port, the capabilities of the port within the Fabric Element
---------------------------------	--

fcFxPortCapFxPortIndex

Type	FcFeFxPortIndex
Provided By	SNMP
Access	R
Description	This object identifies the Fx_Port within the module. This number ranges from 1 to the value of <i>fcFeModulePortCapacity</i> for the associated module. The value remains constant for the identified Fx_Port until the module is re-initialized.

FcFxPortCapFcphVersionHigh

Type	FcphVersion
Provided By	FC2
Access	R

Description	The highest or most recent version of FC-PH that the Fx_Port is capable of supporting. For values see FcFxPortFcphVersionHigh on page 2-69.
-------------	---

FcFxPortCapFcphVersionLow

Type	FcphVersion
Provided By	FC2
Access	R
Description	The lowest or earliest version of FC-PH that the Fx_Port is capable of supporting. For values see FcFxPortFcphVersionHigh on page 2-69.

FcFxPortCapBbCreditMax

Type	FcBbCredit
Provided By	SNMP
Access	R
Description	The maximum number of receive buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames from the attached Nx_Port. This value is fixed at 16.

FcFxPortCapBbCreditMin

Type	FcBbCredit
Provided By	SNMP
Access	R
Description	The minimum number of receive buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames from the attached Nx_Port. This value is fixed at 1.

FcFxPortCapRxDataFieldSizeMax

Type	FcRxDataFieldSize
Provided By	SNMP
Access	R

Description The maximum size in bytes of the Data Field in a frame that the Fx_Port is capable of receiving from its attached Nx_Port.
This value is fixed at 2112.

FcFxPortCapRxDataFieldSizeMin

Type FcRxDataFieldSize
Provided By SNMP
Access R
Description The minimum size in bytes of the Data Field in a frame that the Fx_Port is capable of receiving from its attached Nx_Port.
This value is fixed at 2112.

FcFxPortCapCos

Type FcCosCap
Provided By SNMP
Access R
Description A value indicating the set of Classes of Service that the Fx_Port is capable of supporting.
This value is fixed at CLASS_2 | CLASS_3 (0x0C).

fcFxPortCapIntermix

Type INTEGER
Provided By SNMP
Access R
Description A flag indicating whether or not the Fx_Port is capable of supporting the intermixing of Class 2 and Class 3 frames during a Class 1 connection. This flag is only valid if the port is capable of supporting Class 1 service. The values are defined as follow: yes(1) and no(2).
This value is fixed no(2).

FcFxPortCapStackedConnMode

Type FcStackedConnMode

Provided By	SNMP
Access	R
Description	A value indicating the mode of Stacked Connect request that the Fx_Port is capable of supporting. This value is fixed at none(1).

FcFxPortCapClass2SeqDeliv

Type	INTEGER
Provided By	SNMP
Access	R
Description	A flag indicating whether or not the Fx_Port is capable of supporting Class 2 Sequential Delivery. This value is fixed at yes(1).

FcFxPortCapClass3SeqDeliv

Type	INTEGER
Provided By	SNMP
Access	R
Description	A flag indicating whether or not the Fx_Port is capable of supporting Class 3 Sequential Delivery. This value is fixed at yes(1).

FcFxPortCapHoldTimeMax

Type	MicroSeconds
Provided By	SNMP
Access	R
Description	The maximum holding time (in microseconds) that the Fx_Port is capable of supporting. This value is not supported. It's always zero.

FcFxPortCapHoldTimeMin

Type	MicroSeconds
Provided By	SNMP
Access	R
Description	The minimum holding time (in microseconds) that the Fx_Port is capable of supporting. This value is not supported. It's always zero.

FcFxPortCapBaudRates

Type	Fc0BaudRateCap
Provided By	FPM
Access	R
Description	A value indicating the set of baud rates that the Fx_Port is capable of supporting. This variable has been deprecated and may be implemented for backward compatibility.

FcFxPortCapMedia

Type	Fc0MediaCap
Provided By	FPM
Access	R
Description	A value indicating the set of media that the Fx_Port is capable of supporting.

NOTE: All the counters are 32-bit counters.

Fibre Alliance MIB

Type definitions

FcNameId

Syntax	OCTET STRING (SIZE(8))
Description	Represents the World wide Name (WWN; IEEE 60-bit variety; standard part of T11 definitions for fibre channel) associated with a Fibre Channel (FC) entity.

FcGlobalId

Syntax	OCTET STRING (SIZE(16))
Description	Represents the World wide Name (WWN; IEEE 124-bit variety) associated with a Fibre Channel (FC) entity.

FcEventSeverity

Syntax	INTEGER
Description	<p>The set of values which define the event severity that will be logged by this connectivity unit. Values unknown (1) through debug (9) are essentially self-explanatory; mark (10) means that all messages are logged.</p> <p>The values are defined as follow: unknown (1), emergency (2), alert (3), critical (4), error (5), warning (6), notify (7), info (8), debug (9), mark (10).</p>

FcUnitType

Syntax	INTEGER
Description	The values are defined as unknown (1) – cannot be determined, other (2) – none of the following, hub (3) – passive connectivity unit supporting loop protocol, switch (4) – active connectivity unit supporting multiple protocols, gateway (5) – unit that converts not only the interface but also the frame into another protocol. The assumption is that there is always two gateways connected together.

For example, FC <-> ATM, converter (6) – unit that converts from one interface to another, For example, FC <-> SCSI, hba(7) – host bus adapter, proxyAgent (8) – software proxy-agent, storageDevice (9) – disk, cd, tape, etc, host (10) – host computer, storageSubsystem (11) – raid, library, etc, module (12) – subcomponent of a system, swDriver (13) – software driver, storageAccessDevice (14) – Provides storage management and access for heterogeneous hosts and heterogeneous devices.

FcPortFcClass

Syntax	BITS
Description	Represents the class(es) of service represented on a given port, in a given operational context. The values are defined as follows: unknown (0) classF (1) class1(2) class2 (3) class3 (4) class4 (5) class5 (6) class6 (7)

Connectivity Unit Group

fcConnUnitNumber

Type	INTEGER
Value	1
Access	R
Description	The number of connectivity units present on this system. May be a count of the boards in a chassis or the number of full boxes in a rack.

FcConnURL

Type	DisplayString
------	---------------

Value	http://switch's IP-addr
Access	R
Description	The top-level URL of the system. If it does not exist the value is an empty string. The URL format is implementation dependent and can have keywords embedded that are preceded by a percent sign (e.g.,%USER).

The following are the defined keywords that will be recognized and replaced with data during a launch:

USER	replace with username
PASSWORD	replace with password
GLOBALID	replace with globalid
SERIALNO	replace with serial number

A management application will read this object from the MIB, provide values for any of the keywords listed above that are present in the string, and then use the URL to invoke or launch the program referenced.

FcConnUnitSnsMaxRows

Type	Unsigned32 (Same as Gauge).
Value	The number of the entries of the Name Server Table.
Access	R
Description	The maximum number of rows in the fcConnUnitSnsTable table.

fcConnUnitTable Contains general information on the system's units

***fcConnUnitId**

Type	OCTET STRING
Product Mapping	Switch's WWN.
Access	R
Description	The unique identification for this connectivity unit among those within this proxy domain. The value MUST be unique within the proxy domain because it is the index variable for fcConnUnitTable.

The value assigned to a given connectivity unit SHOULD be persistent across agent and unit resets. It SHOULD be the same as `fcConnUnitGlobalId` if `fcConnUnitGlobalId` is known and stable.

FcConnUnitGlobalId

Type	FcGlobalId
Product Mapping	Switch's WWN.
Access	R
Description	<p>An optional global-scope identifier for this connectivity unit. It MUST be a WWN for this connectivity unit or 16 octets of value zero.</p> <p>WWN formats requiring fewer than 16 octets MUST be extended to 16 octets with trailing zero octets. If a WWN is used for <code>fcConnUnitId</code>, the same WWN MUST be used for <code>fcConnUnitGlobalId</code>.</p> <p>When a non-zero value is provided, it SHOULD be persistent across agent and unit resets. It SHOULD be globally unique. It SHOULD be one of these FC-PH/PH3 formats:</p> <p>IEEE (NAA=1)</p> <p>IEEE Extended (NAA=2)</p> <p>IEEE Registered (NAA=5).</p> <p>IEEE Registered extended (NAA=6).</p> <p>Use of the IEEE formats allows any IEEE-registered vendor to assure global uniqueness independently. The following are some references on IEEE WWN formats:</p> <p>http://standards.ieee.org/regauth/oui/tutorials/fibreformat.html</p> <p>http://standards.ieee.org/regauth/oui/tutorials/fibrecomp_id.html</p> <p>If one or more WWNs are associated with the connectivity unit via other management methods, one of them SHOULD be used for <code>fcConnUnitGlobalId</code>. If there is not a WWN assigned specifically to the connectivity unit, there is some merit, though not a requirement, to using a WWN assigned to (one of) its permanently attached FC/LAN interface(s). This can not risk uniqueness, though.</p> <p>As a counterexample, if your agent runs in a host and the host has an HBA, it is quite possible that agent, host, and HBA will all be distinct connectivity units, so the host and agent can not use the WWN of the</p>

HBA. Another example: If your hub has a built-in Ethernet port, it might be reasonable for the hub to use its LAN address (prefixed with the appropriate NAA) as its `fcConnUnitId`. But if the Ethernet were a replaceable PCCard, the hub should have an independent ID.

FcConnUnitType

Type	FcUnitType
Product Mapping	switch(4)
Access	R
Description	The type of this connectivity unit.

FcConnUnitNumports

Type	Unsigned32
Product Mapping	Number of ports from PROD_CNFG.
Access	R
Description	Number of physical ports in the connectivity unit (internal/embedded, external).

FcConnUnitState

Type	INTEGER
Product Mapping	online and coming-online will indicate online state (2), and offline and going-offline will indicate offline state (3).
Access	R
Description	This object reports the overall state of the connectivity unit. The meaning of all values is essentially self-explanatory. Any of these values may occur with any of the <code>fcConnUnitStatus</code> values. The values are defined as unknown (1), online (2), offline (3).

FcConnUnitStatus

Type	INTEGER
Product Mapping	This value will be mapped from current status of switch in such a way that operational status indicates ok (3), degraded status indicates warning (4), failed status indicate failed (5).

Access	R
Description	<p>This object reports the overall status of the connectivity unit. The warning (4) value means that the connectivity unit needs attention; all other values are essentially self-explanatory. Any of these values may occur with any of the <code>fcConnUnitState</code> values</p> <p>The values are defined as unknown (1), unused (2), ok (3), warning(4), failed (5)</p>

fcConnUnitProduct

Type	SnmpAdminString
Product Mapping	The oem product name.
Access	R
Description	The connectivity unit vendor's product model name.

FcConnUnitSerialNo

Type	SnmpAdminString
Product Mapping	OEM serial number.
Access	R
Description	The serial number identification for this connectivity unit.

FcConnUnitUpTime

Type	TimeTicks
Access	R
Description	The number of centiseconds since the last unit initialization.

FcConnUnitUrl

Type	DisplayString
Product Mapping	Same as <code>fcConnURL</code> .
Access	R/W

Description URL to launch a management application, if applicable. Otherwise empty string. In a standalone unit, this would be the same as the top level URL. This has the same definition as systemURL for keywords.

FcConnUnitDomainId

Type OCTET STRING (SIZE (3))

Product Mapping FFCCXX
XX is the active domainId of the switch.

Access R

Description 24-bit Fibre Channel address ID of this connectivity unit. Following the fibre channel standard, the right-most bit of the right-most octet is for the least significant bit of the address value; the left-most bit of the left-most octet, if needed, is for the most significant bit of the address value. If this value is not applicable, all bits set to 1.

FcConnUnitProxyMaster

Type INTEGER

Product Mapping yes(3)

Access R

Description A value of 'yes' means this is the proxy master unit for a set of managed units. For example, this could be the only unit with a management card in it for a set of units. A standalone unit should return 'yes' for this object. The values are defined as unknown (1), no (2), yes (3).

FcConnUnitPrincipal

Type INTEGER

Access R

Description Whether this connectivity unit is the principal unit within the group of fabric elements. If this value is not applicable, return unknown. The values are defined as unknown (1), no (2), yes (3).

FcConnUnitNumSensors

Type Unsigned32

Product Mapping	The number of sensors.
Access	R
Description	Number of sensors in the <code>fcConnUnitSensorTable</code> .

FcConnUnitNumRevs

Type	Unsigned32
Product Mapping	1
Access	R
Description	The number of revisions in the <code>fcConnUnitRevsTable</code> .

FcConnUnitModuleId

Type	OCTET STRING(SIZE(16))
Product Mapping	return 16 zeros.(currently not supported)
Access	R
Description	This is a unique id, persistent between boots, that can be used to group a set of connectivity units together into a module. The intended use would be to create a connectivity unit with a <code>fcConnUnitType</code> of 'module' to represent a physical or logical group of connectivity units. Then the members of the group would set the value of <code>fcConnUnitId</code> for this 'container' connectivity unit. <code>FcConnUnitModuleId</code> should be zeros if this connectivity unit is not part of a module.

FcConnUnitName

Type	<code>SnmpAdminString</code>
Product Mapping	switch's configured name. Writable and persistent across IPL.
Access	R/W
Description	A name for this connectivity unit. This object value should be persistent between boots.

FcConnUnitInfo

Type	SnmpAdminString
Product Mapping	A textual description of the product. Writable and persistent across IPL.
Access	R/W
Description	Information about this connectivity unit. This object value should be persistent between boots.

FcConnUnitControl

Type	INTEGER
Product Mapping	Always return unknown (1) on read operation. ResetConnUnitWarmStart (4), offlineConnUnit (5), and onlineConnUnit (6) will be supported by PCP. ResetConnUnitWarmStart (4) indicates IPL on the switch is performed. ResetConnUnitColdStart (3) is not supported.
Access	R/W
Description	This object is used to control the addressed connectivity unit. <hr/> NOTE: 'ColdStart' and 'WarmStart' are as defined in mib-2 and are not meant to be a factory reset. <hr/> ResetConnUnitColdStart the addressed unit performs a 'ColdStart' reset. ResetConnUnitWarmStart the addressed unit performs a 'WarmStart' reset. OfflineConnUnit : the addressed unit puts itself into an implementation dependent 'offline' state. In general, if a unit is in an offline state, it cannot be used to perform meaningful Fibre Channel work. OnlineConnUnit the addressed unit puts itself into an implementation dependent 'online' state. In general, if a unit is in an online state, it is capable of performing meaningful Fibre Channel work.

NOTE: Each implementation may chose not to support SNMP Set operations for any or all of these values. For Sets specifying varbinds for instances of this object and values not supported by a given implementation, the agent will return the SNMP WrongValue PDU error code.

The values are defined as follow: unknown (1), invalid (2), resetConnUnitColdStart (3), resetConnUnitWarmStart (4), offlineConnUnit (5), onlineConnUnit (6).

FcConnUnitContact

Type	SnmpAdminString
Product Mapping	Contact information for this connectivity unit. Writable and persistent across IPL.
Access	R/W
Description	Contact information for this connectivity unit. The contact information is intended to facilitate contacting someone in case of problems, questions, etc. (e.g., the help desk internal to a company).

FcConnUnitLocation

Type	SnmpAdminString
Product Mapping	The physical location of the switch. Writable and persistent across IPL.
Access	R/W
Description	Location information for this connectivity unit.

FcConnUnitEventFilter

Type	FcEventSeverity Writable and it's set to the Default value of info(8) after IPL.
Access	R/W This value defines the event severity that will be logged by this connectivity unit. All events of severity less than or equal to <code>fcConnUnitEventFilter</code> are logged in the <code>fcConnUnitEventTable</code> .

FcConnUnitNumEvents

Type	Unsigned32
Product Mapping	Number of events in the <code>fcConnUnitEventTable</code> . It's always ≤ 200 , the maximum size of the event table.
Access	R
Description	Number of events currently in the <code>fcConnUnitEventTable</code> .

FcConnUnitMaxEvents

Type	Unsigned32
Product Mapping	200.
Access	R
Description	Max number of events that can be recorded at any one time in the <code>fcConnUnitEventTable</code> .

FcConnUnitEventCurrID

Type	Unsigned32
Product Mapping	The current event index is used as the last used event id.
Access	R
Description	The last used event id (<code>fcConnUnitEventIndex</code>) recorded in the <code>fcConnUnitEventTable</code> . When no events are presently recorded in the <code>fcConnUnitEventTable</code> , the value of this object MUST be zero.

Firmware Table The revisions table lists the revisions supported by the associated connectivity units.

***fcConnUnitRevsIndex**

Type	Unsigned32
Product Mapping	Not accessible
Access	R
Description	A unique value among all <code>fcConnUnitRevsEntry</code> s with the same value of <code>fcConnUnitId</code> , in the range between 1 and <code>fcConnUnitNumRevs[fcConnUnitId]</code> .

FcConnUnitRevsRevision

Type	SnmpAdminString
Product Mapping	XX.XX.XX (The revision of the switch).
Access	R
Description	A vendor-specific value identifying a revision of a component of the connectivity unit.

FcConnUnitRevsDescription

Type	SnmpAdminString
Product Mapping	Switch Firmware Level
Access	R
Description	Description of a component in the fcConnUnit to which the revision corresponds.

Sensor Table The sensor table lists the sensors(for fan and power supplies) supported by each switch. For each switch, the table will contain a list of all fan and power supply FRU positions, regardless of whether they are installed or not. When a FRU is not installed, the `UnitSensorStatus` for that table entry will be unknown(1). When a power supply or fan FRU is installed or removed, a sensor trap will be sent (if enabled) which contains an index to the appropriate entry in this table, for the affected FRU.

Note that the number of entries in the table does not change when a fan/power supply FRU is installed or removed.

***fcConnUnitSensorIndex**

Type	Unsigned32
Product Mapping	Not assessable.
Access	R
Description	A unique value among all fcConnUnitSensorEntrys with the same value of <code>fcConnUnitId</code> , in the range between 1 and <code>fcConnUnitNumSensors[fcConnUnitId]</code> .

FcConnUnitSensorName

Type	SnmpAdminString
Product Mapping	The module name of the FRU, such as FAN, PWR or THM
Access	R
Description	A textual identification of the sensor intended primarily for operator use.

FcConnUnitSensorStatus

Type	INTEGER
Product Mapping	This value is evaluated from FRU status. The active, backup and update-busy states are mapped to ok(3). And the failed state is mapped to failed(5).
Access	R
Description	The status indicated by the sensor. The values are defined as unknown (1) – the unit cannot determine the status, other (2) – the status does not fit any of the remaining values, ok (3) – indicates good status, warning (4) – indicates the unit needs attention, failed (5) B indicates the unit is non-functional.

FcConnUnitSensorInfo

Type	SnmpAdminString
Product Mapping	The serial number of the FRUs. It's not supported if the module is failed.
Access	R
Description	Miscellaneous static information about the sensor such as its serial number.

FcConnUnitSensorMessage

Type	SnmpAdminString
Product Mapping	The textual description of the FRU status, such as "active" or "failed".
Access	R

Description This describes the status of the sensor as a message. It may also provide more resolution on the sensor indication, for example 'Cover temperature 1503K, above nominal operating range'.

FcConnUnitSensorType

Type INTEGER

Product Mapping fan (4) or power-supply (5)

Access R

Description The type of component being monitored by this sensor. The unknown (1) and other (2) values meanings analogous to those for the `fcConnUnitSensorStatus` object; all other values are essentially self-explanatory.

The values are defined as unknown (1), other (2), battery (3), fan (4), powerSupply (5), transmitter (6), enclosure (7), board (8), and receiver (9).

FcConnUnitSensorCharacteristic

Type INTEGER

Product Mapping Not supported. Always other (2).

Access R

Description The characteristics being monitored by this sensor. The unknown (1) and other (2) values meanings analogous to those for the `fcConnUnitSensorStatus` object; emf (5) refers to electro-magnetic field; all other values are essentially self-explanatory. The values are defined as unknown (1), other (2), temperature (3), pressure (4), emf (5), currentValue (6), airflow (7), frequency (8), and power (9).

The Port Table Generic information on ports for a specific `fcConnUnit`.

fcConnUnitPortIndex

Type Unsigned32

Product Mapping The port index.

Access	R
Description	A unique value among all fcConnUnitPortEntries on this connectivity unit, between 1 and fcConnUnitNumPorts.

FcConnUnitPortType

Type	INTEGER
Product Mapping	<p>If the port is Not installed notPresent(3)</p> <p>else if the Port State is online use the <u>operating</u> Port Type:</p> <p>F_Port = fPort(8) FL_Port = flPort(7) E_Port = ePort(9) H_Port = hubPort(4) B_Port = ePort(9)</p> <p>else use the <u>configured</u> Port Type:</p> <p>Gx_Port = gPort(10) G_Port = gPort(10) Fx_Port = flPort(7) F_Port = fPort(8) E_Port = ePort(9)</p>

Access	R
Description	The port type refers to the protocol active on the port and can take one of the following values:

unknown (1)	cannot be determined
other (2)	none of the following
notPresent (3)	no port
hubPort (4)	hub port
nPort (5)	end port for fabric
lPort (6)	end port for loop

flPort (7)	public loop
fPort (8)	fabric port
ePort (9)	fabric expansion port
gPort (10)	generic fabric port
domainController (1)	domain controller
hubController (12)	hub controller
scsi (13)	parallel SCSI port
escon (14)	escon port
lan (15)	LAN port
wan (16)	WAN port

FcConnUnitPortFCClassCap

Type	FcPortFCClass
Product Mapping	<p>If the port is not installed, <code>fcConnUnitPortFCClassCap = 0</code> else if ES-1000 $H_Port\ 0x18 = class2(0x10) + class3(0x08)$ $B_Port\ 0x58 = classF(0x40) + class2(0x10) + class3(0x08)$ else it depends on the <u>configured</u> Port Type: $Gx_Port\ 0x58 = classF(0x40) + class2(0x10) + class3(0x08)$ $G_Port\ 0x58 = classF(0x40) + class2(0x10) + class3(0x08)$ $Fx_Port\ 0x18 = class2(0x10) + class3(0x08)$ $F_Port\ 0x18 = class2(0x10) + class3(0x08)$ $E_Port\ 0x58 = classF(0x40) + class2(0x10) + class3(0x08)$</p>
Access	R
Description	Bit mask that specifies the classes of service capability of this port. If this object is not applicable, the agent MUST return all bits set to zero.

FcConnUnitPortFCClassOp

Type	FcPortFCClass
------	---------------

Product Mapping	<p>If the port is Not installed <code>fcConnUnitPortFCClassOp = 0</code> else if the Port State is offline <code>fcConnUnitPortFCClassOp = 0</code> else it depends on the <u>operating</u> Port Type: F_Port Use Class of Service specified in Fabric Login FL_Port Use Class of Service specified in one or more Fabric Login's (OR'd together) $E_Port\ 0x58 = classF(0x40) + class2(0x10) + class3(0x08)$</p>
Access	R
Description	Bit mask that specifies the classes of service that are currently operational at this port. If this object is not applicable, the agent MUST return all bits set to zero.

FcConnUnitPortState

Type	INTEGER
Product Mapping	
Access	R
Description	The current state of the port hardware. The bypassed value (4) means that the port is online but is currently being isolated from the loop or fabric for some reason; the other values are essentially self-explanatory. Any value for this object may co-exist with any value for the <code>fcConnUnitPortStatus</code> object. The values are defined as unknown (1), online (2), offline (3), bypassed (4).

FcConnUnitPortStatus

Type	INTEGER
Product Mapping	
Access	R
Description	The current overall protocol status for the port. The warning value (4) means that the port needs attention; the notParticipating value (6) means that protocol is not being processed; the initializing value (7) means that the port is in the process of coming into service; the

bypassed value (8) means that the port has been manually or automatically isolated from the loop or fabric; the other values are essentially self-explanatory. Any value for this object may co-exist with any value for the `fcConnUnitPortState` object.

The values are defined as unknown (1), unused (2), ok (3), warning (4), failure (5), notParticipating (6), initializing (7), bypassed (8).

fcConnUnitPortTransmitterType

Type	INTEGER
Product Mapping	This is mapped from the port technology as not present and serial indicate unknown(1), optical sw1g and optical sw2g indicate shortwave(4). Optical lw1g and optical lw2g indicate longwave(5), copper db9 and copper amp indicate copper(6).
Access	R
Description	The technology of the port transceiver. The values are defined as unknown (1), other (2), unused (3), shortwave (4) longwave (5), copper (6), and scsi (7), longwaveNoOFC (8), shortwaveNoOFC (9), longwaveLED (10),

fcConnUnitPortModuleType

Type	INTEGER
Product Mapping	If the port is not installed, return gbicNotInstalled(8). Otherwise return smallFormFactor(9).
Access	R
Description	<p>The module type of the port connector. This object refers to the hardware implementation of the port. The embedded value (4) means 'fixed' (e.g., oneXnine).</p> <p>The values are defined as unknown (1), other (2), gbic (3), embedded (4), glm(5), gbicSerialId (6), gbicNoSerialId (7), gbicNotInstalled (8), smallFormFactor (9).</p>

fcConnUnitPortWwn

Type	FcNameId
Product Mapping	World Wide Name of the port.
Access	R

Description	The World Wide Name of the port. If applicable, otherwise empty string.
-------------	---

FcConnUnitPortFCId

Type	OCTET STRING (SIZE(3))
Product Mapping	If it's F-port, return fabric address of the node in form of [domain, area, node]. If it's E_Port, return left-adjusted domain ID of the switch.
Access	R
Description	This is the assigned Fibre Channel ID of this port. This value is expected to be a Big Endian value of 24-bits. If this is loop, then it is the ALPA that is connected. If this is an E_Port, then it will only contain the domain ID left justified, zero filled. If this port does not have a Fibre Channel address, return all bits set to 1.

FcConnUnitPortSerialNoSn

Type	SnmpAdminString
Product Mapping	Not applicable.
Access	R
Description	The serial number identification of the unit (e.g., for a GBIC). If this is not applicable, return a zero-length-string.

FcConnUnitPortRevision

Type	SnmpAdminString
Product Mapping	Not applicable.
Access	R
Description	The port revision (e.g., for a GBIC).

FcConnUnitPortVendor

Type	SnmpAdminString
Product Mapping	Not applicable.
Access	R

Description The port vendor (e.g., for a GBIC).

FcConnUnitPortSpeed

Type Gauge32

Product Mapping Return 100000 kilobytes for 1 Gb/s switches and 200000 kilobytes for 2 Gb/s switches.

Access R

Description The speed of the port in kilobytes per second.

FcConnUnitPortControl

Type INTEGER

Product Mapping ResetConnUnitPort(3), offlineConnUnitPort(6), onlineConnUnitPort(7), and portFailure(42501) are the only set-operations are supported. Always return unknown(1) on read.

Access R/W

Description This object is used to control the addressed fcConnUnit's port. Valid commands are:

Unknown (1) and invalid (2) are only used as values that are read.

ResetConnUnitPort (3): If the addressed connectivity unit allows this operation to be performed on this port, the addressed port performs a vendor-specific 'reset' operation. Examples of these operations are: the Link Reset protocol, the Loop Initialization protocol, or a resynchronization occurring between the transceiver in the addressed port to the transceiver that the port is connected to.

BypassConnUnitPort (4): If the addressed connectivity unit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'bypass' operation. Examples of these operations are: transitioning from online to offline, a request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.

UnbypassConnUnitPort (5): If the addressed connectivity unit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'unbypass' operation. Examples of these operations are: the Link Failure protocol, a request (PARTICIPATING) command to the Loop Port state machine, or addition of the port to an arbitrated loop by a hub.

OfflineConnUnitPort (6): If the addressed connectivity unit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'offline' operation. Examples of these operations are: disabling a port's transceiver, the Link Failure protocol, request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.

OnlineConnUnitPort (7): If the addressed connectivity unit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'online' operation. Examples of these operations are: enabling a port's transceiver, the Link Failure protocol, request (PARTICIPATING) command to the Loop Port state machine, or addition of the port from an arbitrated loop by a hub.

NOTE: Each implementation may chose not to support SNMP Set operations for any or all of these values. For Sets specifying varbinds for instances of this object and values not supported by a given implementation, the agent will return the SNMP WrongValue PDU error code.

FcConnUnitPortName

Type	SnmpAdminString
Product Mapping	Port Name
Access	R/W
Description	A string describing the addressed port.

FcConnUnitPortPhysicalNumber

Type	Unsigned32
Product Mapping	Physical port number from 0 to Maximum port number – 1
Access	R
Description	This is the internal port number this port is known by. In many implementations, this should be the same as fcConnUnitPortIndex. Some implementations may have an internal port representation not compatible with the rules for table indexes. In that case, provide the internal representation of this port in this object. This value may also be used in the fcConnUnitLinkPortNumberX or fcConnUnitLinkPortNumberY objects of the fcConnUnitLinkTable.

FcConnUnitPortProtocolCap

(added from Mib3.0)

Type	OCTET STRING (SIZE (2))
Product Mapping	<p>If the port is Not installed</p> <p>$fcConnUnitPortProtocolCap = 0$</p> <p>else if ES-1000</p> <p>H_Port 1 = Loop(1)</p> <p>B_Port 2 = Fabric(2)</p> <p>else it depends on the <u>configured</u> Port Type:</p> <p>Gx_Port 3 = Loop(1) + Fabric(2)</p> <p>G_Port 2 = Fabric(2)</p> <p>Fx_Port 3 = Loop(1) + Fabric(2)</p> <p>F_Port 2 = Fabric(2)</p> <p>E_Port 2 = Fabric(2)</p>
Access	R
Description	<p>Bit mask that specifies the driver level protocol capability of this port.</p> <p>If this is not applicable, return all bits set to zero.</p> <p>The bits have the following definition: unknown – 0, Loop – 1, Fabric – 2,</p> <p>SCSI – 4, TCP/IP - 8, VI – 16, FICON – 32.</p>

FcConnUnitPortProtocolOp

(added from Mib3.0)

Type	OCTET STRING (SIZE (2))
Product Mapping	<p>If the port is Not installed</p> <p>$fcConnUnitPortProtocolOp = 0$</p> <p>else if the Port State is offline</p> <p>$fcConnUnitPortProtocolOp = 0$</p> <p>else it depends on the <u>operating</u> Port Type:</p>

	F_Port 2 = Fabric(2)
	FL_Port 1 = Loop(1)
	E_Port 2 = Fabric(2)
	H_Port 1 = Loop(1)
	B_Port 2 = Fabric(2)
Access	R
Description	Bit mask that specifies the driver level protocol(s) that are currently operational. If this is not applicable, return all bits set to zero. This object has the same definition as <code>fcConnUnitPortProtocolCap</code> .

FcConnUnitPortNodeWwn

	(added from Mib3.0)
Type	FcNameId
Product Mapping	switch WWN
Access	R
Description	The Node World Wide Name of the port if applicable, otherwise all zeros. This should have the same value for a group of related ports. The container is defined as the largest physical entity. For example, all ports on HBAs on a host will have the same Node WWN. All ports on the same storage subsystem will have the same Node WWN.

FcConnUnitPortHWState

	(added from Mib3.0)
Type	INTEGER
Product Mapping	
Access	R
Description	The hardware detected state of the port. The values are defined as follow: unknown (1) failed (2) port failed diagnostics bypassed (3) FCAL bypass loop only,

active (4)	connected to a device
loopback (5)	Port in ext loopback
txfault (6)	Transmitter fault
noMedia (7)	media not installed
linkDown (8)	waiting for activity (rx sync)

The Event Table

The table of connectivity unit events. Errors, warnings, and information should be reported in this table.

*fcConnUnitEventIndex

Type	Unsigned32
Product Mapping	An event index.
Access	R
Description	<p>Each connectivity unit has its own event buffer. As it wraps, it may write over previous events. This object is an index into the buffer. It is recommended that this table be read using 'getNext's to retrieve the initial table.</p> <p>The management application should read the event table at periodic intervals and then determine if any new entries were added by comparing the last known index value with the current highest index value. The management application should then update its copy of the event table.</p> <p>If the read interval is too long, it is possible that there may be events that may not be contained in the agent's internal event buffer. For example, an agent may read events 50-75. At the next read interval, fcConnUnitEventCurrID is 189. If the management app tries to read event index 76, and the agent's internal buffer is 100 entries max, event index 76 will no longer be available.</p> <p>The index value is an incrementing integer starting from one every time there is a table reset. On table reset, all contents are emptied and all indices are set to zero. When an event is added to the table, the event is assigned the next higher integer value than the last item entered into the table.</p> <p>If the index value reaches its maximum value, the next item entered will cause the index value to roll over and start at one again.</p>

FcConnUnitREventTime

Type	DisplayString (SIZE (15))
Product Mapping	The time when the event occurred.
Access	R
Description	This is the real time when the event occurred. It has the following format. DDMMYYYY HHMMSS DD = day number, MM = month number, YYYY = year number, HH = hour number, MM= minute number, SS = seconds number If not applicable, return a NULL string.

FcConnUnitSEventTime

Type	TimeTicks
Product Mapping	Translated from fcConnUnitREventTime.
Access	R
Description	This is the sysuptime timestamp when the event occurred.

FcConnUnitEventSeverity

Type	FcEventSeverity										
Product Mapping	The mapping from switch event severity level to FcEventServerty: <table><tr><td><u>SWITCH</u></td><td><u>MIB</u></td></tr><tr><td>informational</td><td>info(8)</td></tr><tr><td>minor</td><td>error(5)</td></tr><tr><td>major</td><td>critical(4)</td></tr><tr><td>severe</td><td>emergency(2)</td></tr></table>	<u>SWITCH</u>	<u>MIB</u>	informational	info(8)	minor	error(5)	major	critical(4)	severe	emergency(2)
<u>SWITCH</u>	<u>MIB</u>										
informational	info(8)										
minor	error(5)										
major	critical(4)										
severe	emergency(2)										
Access	R										
Description	The event severity level: unknown (1)										

emergency (2)
alert (3)
critical (4)
error (5)
warning (6)
notify (7)
info (8)
debug (9)
mark (10)

FcConnUnitEventType

Type	INTEGER
Product Mapping	Always status(3).
Access	R
Description	The type of this event. The values are defined as follows: unknown (1) other (2) status (3) configuration (4) topology (5)

FcConnUnitEventObject

Type	OBJECT IDENTIFIER
Product Mapping	Only the OID of the <code>fcConnUnit</code> is returned. Other information is not supported.
Access	R
Description	This is used with the <code>fcConnUnitEventType</code> to identify which object the event refers to. It can be the OID of a connectivity unit or of another object like <code>fcConnUnitPortStatus[...]</code>

fcConnUnitEventDescr

Type	SnmpAdminString
Product Mapping	"Reason code XX", XX is the event reason code.
Access	R
Description	The description of the event.

Link Table

The link table is intended to organize and communicate any information the agent which would assist a management application to discover the CONNECTIVITY UNITS in the framework and the TOPOLOGY of their interconnect. That is, the goal is to assist the management application not only to LIST the elements of the framework, but to MAP them.

With this goal, the agent SHOULD include as much as it possesses about any links from its own connectivity units to others, including links among its own units.

An agent SHOULD include partial information about links if it is not able to fully define them. For an entry to be considered to be valid, both the X (local) and the Y (remote) need to have one valid value.

If the agent is able to discover links which do not directly attach to members of its agency and its discovery algorithm gives some assurance the links are recently valid, it MAY include these links.

Link information entered by administrative action MAY be included even if not validated directly if the link has at least one endpoint in this agency, but SHOULD NOT be included otherwise.

A connectivity unit can fill the table in as best it can. One of the methods to fill this in would be to use the RNID ELS (ANSI document 99-422v0). This allows one to query a port for the information needed for the link table.

This table is accessed either directly if the management software has an index value or via GetNexts. The value of the indexes are not required to be contiguous. Each entry created in this table will be assigned an index. This relationship is kept persistent until the entry is removed from the table or the system is reset. The total number of entries are defined by the size of the table.

NOTE: (for E/OS firmware): A reset or firmware load will cause this table to be regenerated from the persistent login database – table indices will most likely be associated with different entries after the reset.

*fcConnUnitLinkIndex

Type	Unsigned32
Product Mapping	A link index.
Access	R
Description	This value is used to create a unique value for each entry in the link table with the same <code>fcConnUnitId</code> . The value can only be reused if it is not currently in use and the value is the next candidate to be used. This value is allowed to wrap at the highest value represented by the number of bits. This value is reset to zero when the system is reset and the first value to be used is 1.

FcConnUnitLinkNodeIdx

Type	OCTET STRING (SIZE(16))
Product Mapping	The WWN of the local <code>fcConnUnit</code> is returned. This information is available for both E and F ports
Access	R
Description	The node WWN of the unit at one end of the link. If the node WWN is unknown and the node is a <code>fcConnUnit</code> in the responding agent then the value of this object MUST BE equal to its <code>fcConnUnitId</code> .

FcConnUnitLinkPortNumberX

Type	Integer32
Product Mapping	The <code>fcConnUnit</code> 's local port number is returned. This information is available for both E and F ports.
Access	R
Description	The port number on the unit specified by <code>fcConnUnitLinkNodeIdx</code> if known, otherwise -1. If the value is non-negative then it will be equal to <code>fcConnUnitPortPhysicalNumber</code> .

FcConnUnitLinkPortWwnX

Type	OCTET STRING
Product Mapping	The local side port WWN on the link. This information is available for both E and F ports.
Access	R
Description	The port WWN of the unit specified by <code>fcConnUnitLinkNodeIdX</code> if known, otherwise 16 octets of binary 0.

FcConnUnitLinkNodeIdY

Type	OCTET STRING (SIZE(16))
Product Mapping	The attached node WWN on the link. This information is available for E_Ports and F ports. E_Port <code>nodeIdY</code> can be retrieved from RNID, F port <code>NodeIdY</code> is supported by FLOGI.
Access	R
Description	The node WWN of the unit at the other end of the link. If the node WWN is unknown and the node is a <code>fcConnUnit</code> in the responding SNMP agent then the value of this object MUST BE equal to its <code>fcConnUnitId</code> .

FcConnUnitLinkPortNumberY

Type	Integer32
Product Mapping	The attached port number on the link. For F port, -1 is returned.
Access	R
Description	The port number on the unit specified by <code>fcConnUnitLinkNodeIdY</code> if known, otherwise -1. If the value is non-negative then it will be equal to <code>fcConnUnitPortPhysicalNumber</code> .

FcConnUnitLinkPortWwnY

Type	OCTET STRING
Product Mapping	The attached port WWN on the link. For E_Ports, returns the WWN of the connected switch.
Access	R

Description	The port WWN on the unit specified by <code>fcConnUnitLinkNodeIdY</code> if known, otherwise 16 octets of binary 0.
-------------	---

FcConnUnitLinkAgentAddressY

Type	OCTET STRING (SIZE(16))
Access	R
Description	The address of an FCMGMT MIB agent for the node identified by <code>fcConnUnitLinkNodeIdY</code> , if known; otherwise 16 octets of binary 0.

FcConnUnitLinkAgentAddressTypeY

Type	Unsigned32
Access	R
Description	If <code>fcConnUnitLinkAgentAddressY</code> is non-zero, then it is a protocol address. <code>FcConnUnitLinkAgentAddressTypeY</code> is the 'address family number' assigned by IANA to identify the address format. (e.g., 1 is Ipv4, 2 is Ipv6).

FcConnUnitLinkAgentPortY

Type	Unsigned32
Access	R
Description	The IP port number for the agent. This is provided in case the agent is at a non-standard SNMP port.

FcConnUnitLinkUnitTypeY

Type	FcUnitType
Product Mapping	If it's E_Port, return switch (4). Otherwise return RNID type Y.
Access	R
Description	Type of the FC connectivity unit as defined in <code>fcConnUnitType</code> .

FcConnUnitLinkConnIdY

Type	OCTET STRING (SIZE(3))
------	------------------------

Product Mapping	For F ports, return Fibre Channel Address. For E_Ports, return left adjusted domainId of the switch.
Access	R
Description	This is the Fibre Channel ID of this port. If the connectivity unit is a switch, this is expected to be a 24-bit Big Endian value. If this is loop, then it is the ALPA that is connected. If this is an E_Port, then it will only contain the domain ID. If not any of those, unknown or cascaded loop, return all bits set to 1.

fcConnUnitPortStatTable - Port statistics

There is one and only one statistics table for each individual port. For all objects in statistics table, if the object is not supported by the conn unit then the high order bit is set to 1 with all other bits set to zero. The high order bit is reserved to indicate if the object if supported or not. All objects start at a value of zero at hardware initialization and continue incrementing till end of 63 bits and then wrap to zero.

***fcConnUnitPortStatIndex**

Type	Unsigned32
Product Mapping	A port number, starting from 1 to maximum number of ports.
Access	R
Description	A unique value among all entries in this table, between 0 and fcConnUnitNumPort[fcConnUnitPortUnitId].

fcConnUnitPortStatCountError

Type	Counter64
Product Mapping	This MIB object counts: address ID errors, CRC errors, delimiter errors, frames too short, invalid transmission words, link failures, primitive sequence errors, signal losses,

synchronization losses.

(Only supports low 32-bits of counter, high 32-bits are set to zero).

Access	R
Description	A count of the errors that have occurred on this port.

FcConnUnitPortStatCountTxObjects

Type	Counter64
Product Mapping	stTxFrames (64-bit counter).
Access	R
Description	The number of frames/packets/Ios/etc that have been transmitted by this port.

NOTE: A Fibre Channel frame starts with SOF and ends with an EOF. FC loop devices should not count frames passed through. This value represents the sum total for all other Tx objects.

FcConnUnitPortStatCountRxObjects

Type	Counter64
Product Mapping	stRxFrames (64-bit counter).
Access	R
Description	The number of frames/packets/Ios/etc that have been received by this port.

NOTE: A Fibre Channel frame starts with an SOF and ends with an EOF. FC loop devices should not count frames passed through. This value represents the sum total for all other Rx objects.

FcConnUnitPortStatCountTxElements

Type	Counter64
Product Mapping	stTxOctets (64-bit counter).
Access	R

Description	The number of octets or bytes that have been transmitted by this port. There is one second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput.
-------------	---

NOTE: for Fibre Channel, ordered sets are not included in the count.

FcConnUnitPortStatCountRxElements

Type	Counter64
Product Mapping	stRxOctets (64-bit counter).
Access	R
Description	The number of octets or bytes that have been received by this port. There is one second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput.

NOTE: For Fibre Channel, ordered sets are not included in the count.

FcConnUnitPortStatCountBBCreditZero

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of transitions in/out of Bbcredit zero state. The other side is not providing any credit.

NOTE: This is a Fibre Channel statistic only.

FcConnUnitPortStatCountInputBuffersFull

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of occurrences when all input buffers of a port were full and outbound buffer-to-buffer credit transitioned to zero. There is no credit to provide to other side.

NOTE: This is a Fibre Channel statistic only.

FcConnUnitPortStatCountFBSYFrames

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of times that FBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. Port can only occur on SOFc1 frames (the frames that establish a connection).

NOTE: This is a Fibre Channel only statistic. This is the sum of all classes.

FcConnUnitPortStatCountPBSYFrames

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of times that PBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if the destination port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection).

NOTE: This is a Fibre Channel only statistic. This is the sum of all classes. If you cannot keep the by class counters, then keep the sum counters.

FcConnUnitPortStatisticCountFRJTFrames

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of times that FRJT was returned to this port as a result of a Frame that was rejected by the fabric.

NOTE: This is the total for all classes and is a Fibre Channel only statistic.

FcConnUnitPortStatCountPRJTFrames

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of times that FRJT was returned to this port as a result of a Frame that was rejected at the destination N_Port.

NOTE: This is the total for all classes and is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass1RxFrames

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of Class 1 Frames received at this port.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass1TxFrames

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of Class 1 Frames transmitted out this port.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass1FBSYFrames

Type	Counter64
Product Mapping	Not supported.

Access	R
Description	Number of times that FBSY was returned to this port as a result of a Class 1 Frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection).

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass1PBSYFrames

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of times that PBSY was returned to this port as a result of a Class 1 Frame that could not be delivered to the other end of the link. This occurs if the destination N_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection).

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass1FRJTFrames

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of times that FRJT was returned to this port as a result of a Class 1 Frame that was rejected by the fabric.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass1PRJTFrames

Type	Counter64
Product Mapping	Not supported.
Access	R

Description Number of times that FRJT was returned to this port as a result of a Class 1 Frame that was rejected at the destination N_Port.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass2RxFrames

Type Counter64

Product Mapping stC2FramesIn (64-bit counter).

Access R

Description Number of Class 2 Frames received at this port.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass2TxFrames

Type Counter64

Product Mapping stC2FramesOut (64-bit counter).

Access R

Description Number of Class 2 Frames transmitted out this port.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass2FBSYFrames

Type Counter64

Product Mapping stC2FabricBusy (Only supports low 32 bits of counter, high 32 bits are set to zero).

Access R

Description Number of times that FBSY was returned to this port as a result of a Class 2 Frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection).

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass2PBSYFrames

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of times that PBSY was returned to this port as a result of a Class 2 Frame that could not be delivered to the other end of the link. This occurs if the destination N_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection).

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass2FRJTFrames

Type	Counter64
Product Mapping	stC2FabricReject (Only supports low 32 bits of counter, high 32 bits are set to zero).
Access	R
Description	Number of times that FRJT was returned to this port as a result of a Class 2 Frame that was rejected by the fabric.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass2PRJTFrames

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of times that FRJT was returned to this port as a result of a Class 2 Frame that was rejected at the destination N_Port.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass3RxFrames

Type	Counter64
Product Mapping	stC3FramesIn (64-bit counter).
Access	R
Description	Number of Class 3 Frames received at this port.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass3TxFrames

Type	Counter64
Product Mapping	stC3FramesOut (64-bit counter).
Access	R
Description	Number of Class 3 Frames transmitted out this port.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountClass3Discards

Type	Counter64
Product Mapping	stC3Discards (64-bit counter).
Access	R
Description	Number of Class 3 Frames that were discarded upon reception at this port. There is no FBSY or FRJT generated for Class 3 Frames. They are simply discarded if they cannot be delivered.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountRxMulticastObjects

Type	Counter64
Product Mapping	Not supported.
Access	R

Description	Number of Multicast Frames or Packets received at this port.
-------------	--

FcConnUnitPortStatCountTxMulticastObjects

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of Multicast Frames or Packets transmitted out this port.

FcConnUnitPortStatCountRxBroadcastObjects

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of Broadcast Frames or Packets received at this port.

FcConnUnitPortStatCountTxBroadcastObjects

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of Broadcast Frames or Packets transmitted out this port. On a Fibre Channel loop, count only OPNr frames generated.

FcConnUnitPortStatCountRxLinkResets

Type	Counter64
Product Mapping	StLinkResetsIn (Only supports low 32 bits of counter, high 32 bits are set to zero).
Access	R
Description	Number of Link resets. This is the number of LRs received.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountTxLinkResets

Type	Counter64
Product Mapping	stLinkResetsOut (Only supports low 32 bits of counter, high 32 bits are set to zero).
Access	R
Description	Number of Link resets. This is the number LRs transmitted.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountNumberLinkResets

Type	Counter64
Product Mapping	StLinkResetsIn + stLinkResetsOut (Only supports low 32 bits of counter, high 32 bits are set to zero).
Access	R
Description	Number of Link resets and LIPs detected at this port. The number times the reset link protocol is initiated. These are the number of the logical resets, a count of the number of primitives.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountRxOfflineSequences

Type	Counter64
Product Mapping	StOlssIn (Only supports low 32 bits of counter, high 32 bits are set to zero).
Access	R
Description	Number of Offline Primitive OLS received at this port.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountTxOfflineSequences

Type	Counter64
------	-----------

Product Mapping StOlssOut (Only supports low 32 bits of counter, high 32 bits are set to zero).

Access R

Description Number of Offline Primitive OLS transmitted by this port.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountNumberOfflineSequences

Type Counter64

Product Mapping stOlssIn + stOlssOut (Only supports low 32 bits of counter, high 32 bits are set to zero).

Access R

Description Number of Offline Primitive sequence received at this port.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountLinkFailures

Type Counter64

Product Mapping stLinkFailures (Only supports low 32 bits of counter, high 32 bits are set to zero).

Access R

Description Number of link failures. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8).

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountInvalidCRC

Type Counter64

Product Mapping stInvalidCrcs (Only supports low 32 bits of counter, high 32 bits are set to zero).

Access R

Description	Number of frames received with invalid CRC. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Loop ports should not count CRC errors passing through when monitoring.
-------------	---

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountInvalidTxWords

Type	Counter64
Product Mapping	stInvalidTxWords (Only supports low 32 bits of counter, high 32 bits are set to zero).

Access	R
--------	---

Description	Number of invalid transmission words received at this port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8).
-------------	---

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountPrimitiveSequenceProtocolErrors

Type	Counter64
Product Mapping	stPrimSeqProtoErrors (Only supports low 32 bits of counter, high 32 bits are set to zero).

Access	R
--------	---

Description	Number of primitive sequence protocol errors detected at this port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8).
-------------	---

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountLossOfSignal

Type	Counter64
Product Mapping	stSigLosses (Only supports low 32 bits of counter, high 32 bits are set to zero).

Access	R
--------	---

Description	Number of instances of signal loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8).
-------------	--

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountLossofSynchronization

Type	Counter64
Product Mapping	stSyncLosses (Only supports low 32 bits of counter, high 32 bits are set to zero).
Access	R
Description	Number of instances of synchronization loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8).

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountInvalidOrderedSets

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of invalid ordered sets received at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8).

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountFramesTooLong

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of frames received at this port where the frame length was greater than what was agreed to in FLOGI/PLOGI. This could be caused by losing the end of frame delimiter.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountFramesTruncated

Type	Counter64
Product Mapping	stFramesTooShort (Only supports low 32 bits of counter, high 32 bits are set to zero).
Access	R
Description	Number of frames received at this port where the frame length was less than the minimum indicated by the frame header – normally 24 bytes, but it could be more if the DFCTL field indicates an optional header should have been present.

NOTE: This is a Fibre Channel only statistic

fcConnUnitPortStatCountAddressErrors

Type	Counter64
Product Mapping	stAddrIDErrors (Only supports low 32 bits of counter, high 32 bits are set to zero).
Access	R
Description	Number of frames received with unknown addressing. E.g. unknown SID or DID. The SID or DID is not known to the routing algorithm.

NOTE: This is a Fibre Channel only statistic.

FcConnUnitPortStatCountDelimiterErrors

Type	Counter64
Product Mapping	stDelimiterErrors (Only supports low 32 bits of counter, high 32 bits are set to zero).
Access	R
Description	Number of invalid frame delimiters received at this port. An example is a frame with a class 2 start and a class 3 at the end.

NOTE: This is a Fibre Channel only statistic.

fcConnUnitPortStatCountEncodingDisparityErrors

Type	Counter64
Product Mapping	Not supported.
Access	R
Description	Number of disparity errors received at this port.

NOTE: This is a Fibre Channel only statistic.

Name Server Table This table is accessed either directly (if the management software has an index value) or via GetNexts. The value of the indices need not be contiguous. Each entry created in this table will be assigned an index. This relationship is kept persistent until the entry is removed from the table or the system is reset. The total number of entries is defined by the size of the table.

fcConnUnitSnsPortIndex

Type	Unsigned32 (Same as Gauge)
Product Mapping	A port number, starting from 1 to maximum number of ports.
Access	R
Description	The physical port number of this SNS table entry. Each physical port has an SNS table with 1-n entries indexed by <code>fcConnUnitSnsPortIdentifier</code> (port address).

fcConnUnitSnsPortIdentifier

Type	FcAddressId
Product Mapping	3 bytes FcAddress in the least significant bytes.
Access	R
Description	The Port Identifier for this entry in the SNS table.

fcConnUnitSnsPortName

Type	FcNaneId
------	----------

Product Mapping	Port WWN Name
Access	R
Description	The Port WWN Name for this entry in the SNS table.

fcConnUnitSnsNodeName

Type	FcNameId
Product Mapping	Node Name.
Access	R
Description	The Node Name for this entry in the SNS table.

fcConnUnitSnsClassOfSvc

Type	OCTET STRING (SIZE) (1))
Product Mapping	Class of Service that matches the FC class service convention used in name server.
Access	R
Description	The classes of service offered by this entry in the SNS table.

fcConnUnitSnsNodeIPAddress

Type	OCTET STRING (SIZE) (16))
Product Mapping	Node IP address.
Access	R
Description	The Ipv6 formatted address of the Node for this entry in the SNS table. In order for this data to be present, IP address must have been registered with the switch.

fcConnUnitSnsProcAssoc

Type	OCTET STRING (SIZE) (8))
Product Mapping	Process Associator.
Access	R
Description	The Process Associator for this entry in the SNS table. See FC-PH sec. 19.4.

fcConnUnitSnsFC4Type

Type	OCTET STRING (SIZE) (32))
Product Mapping	FC4 type.
Access	R
Description	The FC-4 Types supported by this entry in the SNS table. Bitmap of FC-4 types supported. See FC-GS2 table 27.

fcConnUnitSnsPortType

Type	OCTET STRING (SIZE) (1))
Product Mapping	Port type.
Access	R
Description	The Port Type of this entry in the SNS table. See FC-GS2 table 5.

fcConnUnitSnsPortIPAddress

Type	OCTET STRING (SIZE) (16))
Product Mapping	Port IP Address.
Access	R
Description	In order for this data to be present, IP address must have been registered with the switch. See FC-GS2 sec 12.4.5.

fcConnUnitSnsFabricPortName

Type	FcNameId
Product Mapping	Fabric Port Name.
Access	R
Description	The Fabric Port name of this entry in the SNS table.

fcConnUnitSnsHardAddress

Type	FcGlobalId
Product Mapping	Bytes address from name server in the least significant bytes.
Access	R
Description	The Hard ALPA of this entry in the SNS table. This address is device selected, not dynamically assigned.

fcConnUnitSnsSymbolicPortName

Type	DisplayString (SIZE (0..79))
Product Mapping	Symbolic port name.
Access	R
Description	The Symbolic Port Name of this entry in the SNS table.

fcConnUnitSnsSymbolicNodeName

Type	DisplayString (SIZE (0..79))
Product Mapping	Symbolic node name.
Access	R
Description	The Symbolic Node Name of this entry in the SNS table.

SNMP Trap Registration Group

fcTrapMaxClients

Type	Unsigned32
Product Mapping	The maximum number of SNMP trap recipients can be supported in the system.
Access	R
Description	The maximum number of SNMP trap recipients supported by the connectivity unit.

FcTrapClientCount

Type	Unsigned32
------	------------

Product Mapping	The current number of trap recipients.
Access	R
Description	The current number of rows in the trap table.

TrapRegTable

A table containing a row for each IP address/port number that traps will be sent to.

*fcTrapRegIpAddress

Type	IpAddress
Product Mapping	Trap recipient's IP address.
Access	R/C
Description	The IP address of a client registered for traps.

*fcTrapRegPort

Type	Unsigned32
Product Mapping	UDP port.
Access	R/C
Description	The UDP port to send traps to for this host. Normally this would be the standard trap port (UDP/162).

FcTrapRegFilter

Type	FcEventSeverity
Product Mapping	The severity filter. (This information is not exposed in the SNMP dialog)
Access	R/C
Description	This value defines the trap severity filter for this trap host. The fcConnUnit will send to the designated target entity traps that have a severity level less than or equal to this value.

FcTrapRegRowState

Type	RowStatus
------	-----------

Product Mapping	Row status.
Access	R/C
Description	<p>Specifies the operational status of the row.</p> <p>A RowStatus object may take any of six defined values:</p> <p>active (1): traps may be sent as specified in this row. A management application may change the value of any objects in the row when the status is active.</p> <p>notInService (2): traps will not be sent using this row.</p> <p>notReady (3): the conceptual row exists in the agent, but is missing information necessary to send traps (i.e., if any of the other objects in the row are not present or contain invalid values). This value may not be supplied by a management application.</p> <p>createAndGo (4): supplied by a management application wishing to create a new instance of a conceptual row, supplying valid values for the all the other objects in the row, and have its status automatically set to active, making it available for use in sending traps.</p> <p>createAndWait (5): supplied by a management application wishing to create a new instance of a conceptual row but not make it available for use in sending traps at that time.</p> <p>destroy (6): supplied by a management application wishing to delete an existing conceptual row.</p>

Trap Types

fcConnUnitStatusChange

Type Number	1
Product Mapping	Generated when the switch's online status or operational status changes.
OID and Value	".1.3.6.1.2.1.8888.1.1.3.1.6" + unitId fcConnUnitStatus, ".1.3.6.1.2.1.8888.1.1.3.1.5" + unitId fcConnUnitState
Description	<p>The overall status of the connectivity unit has changed.</p> <p>Recommended severity level (for filtering): alert.</p>

fcConnUnitDeletedTrap

Type Number	2
Product Mapping	Not supported on the connUnit.
OID and Value	N/A
Description	An fcConnUnit has been deleted from this agent. Recommended severity level (for filtering): warning.

fcConnUnitEventTrap

Type Number	3
Product Mapping	Generated when a new event is generated.
OID and Value	".1.3.6.1.2.1.8888.1.1.7.1.1" + unitId fcConnUnitEventIndex, ".1.3.6.1.2.1.8888.1.1.7.1.5" + unitId fcConnUnitEventType, ".1.3.6.1.2.1.8888.1.1.7.1.6" + unitId fcConnUnitEventObject, ".1.3.6.1.2.1.8888.1.1.7.1.7" + unitId fcConnUnitEventDescr
Description	An event has been generated by the connectivity unit. Recommended severity level (for filtering): info.

fcConnUnitSensorStatusChange

Type Number	4
Product Mapping	Generated when one of the fans or power supply status is changed.
OID and Value	".1.3.6.1.2.1.8888.1.1.5.1.3" + unitId + sensor_nbr fcConnUnitSensorState
Description	The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.

fcConnUnitPortStatusChange

Type Number	5
Product Mapping	Generated when a port state or status is changed.

OID and Value	<code>".1.3.6.1.2.1.8888.1.1.6.1.6" + unitId + port_nbr</code> <code>fcConnUnitPortStatus,</code> <code>".1.3.6.1.2.1.8888.1.1.6.1.5" + unitId + port_nbr</code> <code>fcConnUnitPortState</code>
Description	The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.

FA MIB

Changed snsPortIndex to counter32 (8/12/02)

Modified UTCTime from 13 digits (200XXXXX0000Z) to 11 digits (0XXXXX0000Z).

Changed the Syntax of fcConnUnitPortSpeed from gauge32 to Unsigned32.

Rename the MIB from fcmgmt.mib to fa.mib

May 14, 2002 lxx

FIBRE-CHANNEL-MGMT-MIB DEFINITIONS ::= BEGIN

IMPORTS

OBJECT-TYPE,

NOTIFICATION-TYPE,

MODULE-IDENTITY,

Integer32,

Unsigned32,

Counter32,

Counter64,

IpAddress,

TimeTicks,

```
mib-2
    FROM SNMPv2-SMI
TEXTUAL-CONVENTION,
    DisplayString,
    RowStatus
    FROM SNMPv2-TC
MODULE-COMPLIANCE,
OBJECT-GROUP,
NOTIFICATION-GROUP
    FROM SNMPv2-CONF
    SnmpAdminString
    FROM SNMP-FRAMEWORK-MIB;
fcMgmtMIB MODULE-IDENTITY
LAST-UPDATED 0105080000Z
ORGANIZATION IETF IPFC Working Group
CONTACT-INFO S. Blumenau
              EMC Corporation
              171 South Street
              Hopkinton, MA 01748-9103 U.S.A
              Tel: +1 508 435 1000
              Fax: +1 508 435 4657
              Email: blumenau_steven@emc.com"
Description  The fibre channel management MIB module.
REVISION 0105080000Z
Description  Changed fcConnUnitId and fcConnUnitPortIndex
              from not-accessible to read-only

REVISION 0012040000Z
```


Description Made the following edits:

Used FcPortFCClass as the type for the fcConnUnitPortFCClassCap and fcConnUnitPortFCClassOp objects in the FcConnUnitPortEntry SEQUENCE statement.

Used fcConnUnitGlobalId instead of fcConnUnitId as notification objects in the fcConnUnitDeletedTrap and fcConnUnitEventTrap NOTIFICATION-TYPE macros.

REVISION 0011260000Z

Description The goal of this version was to re-write into SMIV2.

REVISION 0004120000Z

Description Initial revision, published as RFC XXXX.

::= { mib-2 8888 } -- TO BE ASSIGNED

fcMgmtNotifications	OBJECT IDENTIFIER	::= { fcMgmtMIB 0 }
fcMgmtObjects	OBJECT IDENTIFIER	::= { fcMgmtMIB 1 }
fcMgmtConformance	OBJECT IDENTIFIER	::= { fcMgmtMIB 2 }
fcMgmtConfig	OBJECT IDENTIFIER	::= { fcMgmtObjects 1 }
fcMgmtNotifyFilter	OBJECT IDENTIFIER	::= { fcMgmtObjects 2 }
fcMgmtStatistics	OBJECT IDENTIFIER	::= { fcMgmtObjects 3 }
fcMgmtSNS	OBJECT IDENTIFIER	::= { fcMgmtObjects 4 }
fcMgmtCompliances	OBJECT IDENTIFIER	::= { fcMgmtConformance 1 }
fcMgmtGroups	OBJECT IDENTIFIER	::= { fcMgmtConformance 2 }

Textual conventions for this MIB

FcNameId

Status	current
Description	Represents the Worldwide Name (WWN; IEEE 60-bit variety; standard part of T11 definitions for fibre channel) associated with a Fibre Channel (FC) entity.
Syntax	OCTET STRING (SIZE (8))

FcGlobalId

Status	current
Description	Represents the Worldwide Name (WWN; IEEE 124-bit variety) associated with a Fibre Channel (FC) entity.
Syntax	OCTET STRING (SIZE(16))

FcEventSeverity

Status	current
Description	The set of values which define the event severity that will be logged by this connectivity unit. Values unknown (1) through debug (9) are essentially self-explanatory; mark (10) means that all messages are logged.
Syntax	INTEGER { unknown (1), emergency (2), alert (3), critical (4), error (5), warning (6), notify (7), info (8), debug (9),

mark (10) }

FcUnitType

Status	current	
Description		
	unknown (1)	cannot be determined
	other (2)	none of the following
	hub (3)	passive connectivity unit supporting loop protocol.
	switch (4)	active connectivity unit supporting multiple protocols.
	gateway (5)	unit that converts not only the interface but also encapsulates the frame into another protocol. The assumption is that there are always two gateways connected together. For example, FC <-> ATM.
	converter (6)	unit that converts from one interface to another. For example, FC <-> SCSI.
	hba (7)	host bus adapter
	proxyAgent (8)	software proxy-agent
	storageDevice (9)	disk,cd,tape,etc
	host (10)	host computer
	storageSubsystem (11)	raid, library, etc
	module (12)	subcomponent of a system
	swDriver (13)	software driver
	storageAccessDevice (14)	provides storage management and access for heterogeneous hosts and heterogeneous devices.
Syntax	INTEGER {	
	unknown(1),	
	other(2),	
	hub(3),	
	switch(4),	

```
gateway(5),  
converter(6),  
hba(7),  
proxyAgent(8),  
storageDevice(9),  
host(10),  
storageSubsystem(11),  
module(12),  
swDriver(13),  
storageAccessDevice(14) }
```

FcPortFCClass

Status	current
Description	Represents the class(es) of service represented on a given port, in a given operational context.
Syntax	<pre>BITS { unknown (0), classF (1), class1 (2), class2 (3), class3 (4), class4 (5), class5 (6), class6 (7) }</pre>

Connectivity unit group

fcConnUnitNumber

Syntax	Unsigned32
Max-Access	read-only

Status	current
Description	The number of connectivity units present on this system. May be a count of the boards in a chassis or the number of full boxes in a rack.
Sequence	::= { fcMgmtConfig 1 }

fcConnURL

Syntax	DisplayString								
Max-Access	read-only								
Status	current								
Description	<p>The top-level URL of the system. If it does not exist the value is an empty string. The URL format is implementation dependent and can have keywords embedded that are preceeded by a percent sign (eg, %USER).</p> <p>The following are the defined keywords that will be recognized and replaced with data during a launch:</p> <table><tr><td>USER</td><td>- replace with username</td></tr><tr><td>PASSWORD</td><td>- replace with password</td></tr><tr><td>GLOBALID</td><td>- replace with globalid</td></tr><tr><td>SERIALNO</td><td>- replace with serial number</td></tr></table> <p>A management application will read this object from the MIB, provide values for any of the keywords listed above that are present in the string, and then use the URL to invoke or launch the program referenced.</p>	USER	- replace with username	PASSWORD	- replace with password	GLOBALID	- replace with globalid	SERIALNO	- replace with serial number
USER	- replace with username								
PASSWORD	- replace with password								
GLOBALID	- replace with globalid								
SERIALNO	- replace with serial number								
Sequence	::= { fcMgmtConfig 2 }								

Connectivity table	The connectivity table contains general information on the system's connectivity units.
--------------------	---

fcConnUnitTable

Syntax	SEQUENCE OF FcConnUnitEntry
Max-Access	not-accessible
Status	current

Description The connectivity table contains general information on the system's units. The number of entries is given by the value of `fcConnUnitNumber`. It is 1 for stand-alone systems.

Sequence ::= { `fcMgmtConfig 3` }

fcConnUnitEntry

Syntax `FcConnUnitEntry`

Max-Access not-accessible

Status current

Description A connectivity unit entry containing objects for a particular unit.
INDEX { `fcConnUnitId` }

Sequence ::= { `fcConnUnitTable 1` }

`FcConnUnitEntry` ::= SEQUENCE {
 `fcConnUnitId` OCTET STRING,
 `fcConnUnitGlobalId` FcGlobalId,
 `fcConnUnitType` FcUnitType,
 `fcConnUnitNumPorts` Unsigned32,
 `fcConnUnitState` INTEGER,
 `fcConnUnitStatus` INTEGER,
 `fcConnUnitProduct` SnmpAdminString,
 `fcConnUnitSerialNo` SnmpAdminString,
 `fcConnUnitUpTime` TimeTicks,
 `fcConnUnitUrl` DisplayString,
 `fcConnUnitDomainId` OCTET STRING,
 `fcConnUnitProxyMaster` INTEGER,
 `fcConnUnitPrincipal` INTEGER,
 `fcConnUnitNumSensors` Unsigned32,
 `fcConnUnitNumRevs` Unsigned32,
 `fcConnUnitModuleId` OCTET STRING,
 `fcConnUnitName` SnmpAdminString,
 `fcConnUnitInfo` SnmpAdminString,
 `fcConnUnitControl` INTEGER,

fcConnUnitContact	SnmpAdminString,
fcConnUnitLocation	SnmpAdminString,
fcConnUnitEventFilter	FcEventSeverity,
fcConnUnitNumEvents	Unsigned32,
fcConnUnitMaxEvents	Unsigned32,
fcConnUnitEventCurrID	Unsigned32 }

fcConnUnitId

Syntax	OCTET STRING (SIZE (16))
Max-Access	read-only
Status	current
Description	<p>The unique identification for this connectivity unit among those within this proxy domain. The value MUST be unique within the proxy domain because it is the index variable for fcConnUnitTable.</p> <p>The value assigned to a given conectivity unit SHOULD be persistent across agent and unit resets. It SHOULD be the same as fcConnUnitGlobalId if fcConnUnitGlobalId is known and stable.</p>
Sequence	::= { fcConnUnitEntry 1 }

fcConnUnitGlobalId

Syntax	FcGlobalId
Max-Access	read-only
Status	current
Description	<p>An optional global-scope identifier for this connectivity unit. It MUST be a WWN for this connectivity unit or 16 octets of value zero.</p> <p>WWN formats requiring fewer than 16 octets MUST be extended to 16 octets with trailing zero octets. If a WWN is used for fcConnUnitId, the same WWN MUST be used for fcConnUnitGlobalId.</p> <p>When a non-zero value is provided, it SHOULD be persistent across agent and unit resets. It SHOULD be globally unique. It SHOULD be one of these FC-PH/ PH3 formats:</p> <p>IEEE (NAA=1)</p>

IEEE Extended (NAA=2)

IEEE Registered (NAA=5).

IEEE Registered extended (NAA=6).

Use of the IEEE formats allows any IEEE-registered vendor to assure global uniqueness independently. The following are some references on IEEE WWN formats:

<http://standards.ieee.org/regauth/oui/tutorials/fibreformat.html>
http://standards.ieee.org/regauth/oui/tutorials/fibrecomp_id.html

If one or more WWNs are associated with the connectivity unit via other management methods, one of them SHOULD be used for fcConnUnitGlobalId. If there is not a WWN assigned specifically to the connectivity unit, there is some merit, though not a requirement, to using a WWN assigned to (one of) its permanently attached FC/LAN interface(s). This can not risk uniqueness, though. As a counterexample, if your agent runs in a host and the host has an HBA, it is quite possible that agent, host, and HBA will all be distinct connectivity units, so the host and agent can not use the WWN of the HBA.

Another example: If your hub has a built-in Ethernet port, it might be reasonable for the hub to use its LAN address (prefixed with the appropriate NAA) as its fcConnUnitId. But if the Ethernet were a replaceable PCCard, the hub should have an independent ID.

Sequence	::= { fcConnUnitEntry 2 }
<hr/>	
fcConnUnitType	
Syntax	FcUnitType
Max-Access	read-only
Status	current
Description	The type of this connectivity unit.
Sequence	::= { fcConnUnitEntry 3 }

<hr/>	
fcConnUnitNumPorts	
Syntax	Unsigned32

Max-Access	read-only
Status	current
Description	The number of physical ports in the connectivity unit (internal/embedded, external).
Sequence	::= { fcConnUnitEntry 4 }

fcConnUnitState

Syntax	INTEGER { unknown(1), online(2), offline(3) }
Max-Access	read-only
Status	current
Description	This object reports the overall state of the connectivity unit. The meaning of all values is essentially self-explanatory. Any of these values may occur with any of the fcConnUnitStatus values.
Sequence	::= { fcConnUnitEntry 5 }

fcConnUnitStatus

Syntax	INTEGER { unknown(1), unused(2), ok(3), warning(4), failed(5)}
Max-Access	read-only
Status	current
Description	This object reports the overall status of the connectivity unit. The warning (4) value means that the connectivity unit needs attention; all other values are essentially self-explanatory. Any of these values may occur with any of the fcConnUnitState values.
Sequence	::= { fcConnUnitEntry 6 }

fcConnUnitProduct

Syntax	SnmpAdminString
--------	-----------------

Max-Access	read-only
Status	current
Description	The connectivity unit vendor's product model name.
Sequence	::= { fcConnUnitEntry 7 }

fcConnUnitSerialNo

Syntax	SnmpAdminString
Max-Access	read-only
Status	current
Description	The serial number identification for this connectivity unit.
Sequence	::= { fcConnUnitEntry 8 }

fcConnUnitUpTime

Syntax	TimeTicks
Max-Access	read-only
Status	current
Description	The number of centiseconds since the last unit initialization.
Sequence	::= { fcConnUnitEntry 9 }

fcConnUnitUrl

Syntax	DisplayString
Max-Access	read-only
Status	current
Description	URL to launch a management application, if applicable. Otherwise empty string. In a standalone unit, this would be the same as the top-level URL. This has the same definition as fcConnURL for keywords.
Sequence	::= { fcConnUnitEntry 10 }

fcConnUnitDomainId

Syntax	OCTET STRING (SIZE(3))
--------	------------------------

Max-Access	read-only
Status	current
Description	24 bit Fibre Channel address ID of this connectivity unit. Following the fibre channel standard, the right-most bit of the right-most octet is for the least significant bit of the address value; the left-most bit of the left-most octet, if needed, is for the most significant bit of the address value. If this value is not applicable, all bits set to 1.
Sequence	::= { fcConnUnitEntry 11 }

fcConnUnitProxyMaster

Syntax	INTEGER { unknown(1), no(2), yes(3) }
Max-Access	read-only
Status	current
Description	A value of 'yes' means this is the proxy master unit for a set of managed units. For example, this could be the only unit with a management card in it for a set of units. A standalone unit should return 'yes' for this object.
Sequence	::= { fcConnUnitEntry 12 }

fcConnUnitPrincipal

Syntax	INTEGER { unknown(1), no(2), yes(3) }
Max-Access	read-only
Status	current
Description	Whether this connectivity unit is the principal unit within the group of fabric elements. If this value is not applicable, return unknown.
Sequence	::= { fcConnUnitEntry 13 }

fcConnUnitNumSensors

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	Number of sensors in the fcConnUnitSensorTable.

Sequence ::= { fcConnUnitEntry 14 }

fcConnUnitNumRevs

Syntax Unsigned32
 Max-Access read-only
 Status current
 Description The number of revisions in the fcConnUnitRevsTable.
 Sequence ::= { fcConnUnitEntry 15 }

fcConnUnitModuleId

Syntax OCTET STRING (SIZE(16))
 Max-Access read-only
 Status current
 Description This is a unique id, persistent between boots, that can be used to group a set of connectivity units together into a module. The intended use would be to create a connectivity unit with a fcConnUnitType of 'module' to represent a physical or logical group of connectivity units. Then the members of the group would set the value of fcConnUnitId for this 'container' connectivity unit. fcConnUnitModuleId should be zeros if this connectivity unit is not part of a module.
 Sequence ::= { fcConnUnitEntry 16 }

fcConnUnitName

Syntax SnmpAdminString
 Max-Access read-write
 Status current
 Description A name for this connectivity unit. This object value should be persistent between boots.
 Sequence ::= { fcConnUnitEntry 17 }

fcConnUnitInfo

Syntax	SnmpAdminString
Max-Access	read-write
Status	current
Description	Information about this connectivity unit. This object value should be persistent between boots.
Sequence	::= { fcConnUnitEntry 18 }

fcConnUnitControl

Syntax	INTEGER { unknown(1), invalid(2), resetConnUnitColdStart(3), resetConnUnitWarmStart(4), offlineConnUnit(5), onlineConnUnit(6) }
Max-Access	read-write
Status	current
Description	<p>This object is used to control the addressed connectivity unit.</p> <p>NOTE: 'ColdStart' and 'WarmStart' are as defined in mib-2 and are not meant to be a factory reset.</p> <p>resetConnUnitColdStart: the addressed unit performs a 'ColdStart' reset.</p>

resetConnUnitWarmStart: the addressed unit performs a 'WarmStart' reset.

offlineConnUnit: the addressed unit puts itself into an implementation dependant 'offline' state. In general, if a unit is in an offline state, it cannot be used to perform meaningful Fibre Channel work.

onlineConnUnit: the addressed unit puts itself into an implementation dependant 'online' state. In general, if a unit is in an online state, it is capable of performing meaningful Fibre Channel work.

NOTE: Each implementation may chose not to support SNMP Set operations for any or all of these values. For Sets specifying varbinds for instances of this object and values not supported by a given implementation, the agent will return the SNMP WrongValue PDU error code.

Sequence ::= { fcConnUnitEntry 19 }

fcConnUnitContact

Syntax SnmpAdminString

Max-Access read-write

Status current

Description Contact information for this connectivity unit. The contact information is intended to facilitate contacting someone in case of problems, questions, etc. (e.g., the a help desk internal to a company).

Sequence ::= { fcConnUnitEntry 20 }

fcConnUnitLocation

Syntax SnmpAdminString

Max-Access read-write

Status current

Description Location information for this connectivity unit.

Sequence ::= { fcConnUnitEntry 21 }

fcConnUnitEventFilter

Syntax	FcEventSeverity
Max-Access	read-write
Status	current
Description	This value defines the event severity that will be logged by this connectivity unit. All events of severity less than or equal to fcConnUnitEventFilter are logged in the fcConnUnitEventTable.
Sequence	::= { fcConnUnitEntry 22 }

fcConnUnitNumEvents

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	Number of events currently in the fcConnUnitEventTable.
Sequence	::= { fcConnUnitEntry 23 }

fcConnUnitMaxEvents

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	Max number of events that can be recorded at any one time in the fcConnUnitEventTable.
Sequence	::= { fcConnUnitEntry 24 }

fcConnUnitEventCurrID

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	The last used event ID (fcConnUnitEventId) recorded in the fcConnUnitEventTable. When no events are presently recorded in the fcConnUnitEventTable, the value of this object MUST be zero.

Sequence ::= { fcConnUnitEntry 25 }

Revisions Table The revisions table lists the revisions supported by the associated connectivity units.

fcConnUnitRevsTable

Syntax	SEQUENCE OF FcConnUnitRevsEntry
Max-Access	not-accessible
Status	current
Description	Table of the revisions of components (e.g., firmware, hardware, etc.) supported by the connectivity units managed by this agent.
Sequence	::= { fcMgmtConfig 4 }

fcConnUnitRevsEntry

Syntax	FcConnUnitRevsEntry
Max-Access	not-accessible
Status	current
Description	A row in the fcConnUnitRevsTable. INDEX { fcConnUnitId, fcConnUnitRevsIndex }
Sequence	::= { fcConnUnitRevsTable 1 } FcConnUnitRevsEntry ::= SEQUENCE { fcConnUnitRevsIndex Unsigned32, fcConnUnitRevsRevision SnmpAdminString, fcConnUnitRevsDescription SnmpAdminString }

fcConnUnitRevsIndex

Syntax	Unsigned32
Max-Access	not-accessible
Status	current

Description	A unique value among all fcConnUnitRevsEntrys with the same value of fcConnUnitId, in the range between 1 and fcConnUnitNumRevs[fcConnUnitId].
Sequence	::= { fcConnUnitRevsEntry 1 }

fcConnUnitRevsRevision

Syntax	SnmpAdminString
Max-Access	read-only
Status	current
Description	A vendor-specific value identifying a revision of a component of the connectivity unit.
Sequence	::= { fcConnUnitRevsEntry 2 }

fcConnUnitRevsDescription

Syntax	SnmpAdminString
Max-Access	read-only
Status	current
Description	Description of a component in the ConnUnit to which the revision corresponds.
Sequence	::= { fcConnUnitRevsEntry 3 }
	The sensor table list the sensors supported by each connectivity unit.

fcConnUnitSensorTable

Syntax	SEQUENCE OF FcConnUnitSensorEntry
Max-Access	not-accessible
Status	current
Description	Table of the sensors supported by each connectivity unit.
Sequence	::= { fcMgmtConfig 5 }

fcConnUnitSensorEntry

Syntax	FcConnUnitSensorEntry
Max-Access	not-accessible
Status	current
Description	Each entry contains the information for a specific sensor. INDEX { fcConnUnitId, fcConnUnitSensorIndex }
Sequence	::= { fcConnUnitSensorTable 1 } FcConnUnitSensorEntry ::= SEQUENCE { fcConnUnitSensorIndex Unsigned32, fcConnUnitSensorName SnmpAdminString, fcConnUnitSensorStatus INTEGER, fcConnUnitSensorInfo SnmpAdminString, fcConnUnitSensorMessage SnmpAdminString, fcConnUnitSensorType INTEGER, fcConnUnitSensorCharacteristic INTEGER }

fcConnUnitSensorIndex

Syntax	Unsigned32
Max-Access	not-accessible
Status	current
Description	A unique value among all fcConnUnitSensorEntrys with the same value of fcConnUnitId, in the range between 1 and fcConnUnitNumSensors[fcConnUnitId].
Sequence	::= { fcConnUnitSensorEntry 1 }

fcConnUnitSensorName

Syntax	SnmpAdminString
Max-Access	read-only
Status	current

Description	A textual identification of the sensor intended primarily for operator use.
Sequence	::= { fcConnUnitSensorEntry 2 }

fcConnUnitSensorStatus

Syntax	INTEGER { unknown(1), other(2), ok(3), warning(4), failed(5) }
Max-Access	read-only
Status	current
Description	The status indicated by the sensor. unknown (1) the unit cannot determine the status other (2) the status does not fit any of the remaining values ok (3) indicates good status warning (4) indicates the unit needs attention failed (5) indicates the unit is non-functional"
Sequence	::= { fcConnUnitSensorEntry 3 }

fcConnUnitSensorInfo

Syntax	SnmpAdminString
Max-Access	read-only
Status	current
Description	Miscellaneous static information about the sensor such as its serial number.
Sequence	::= { fcConnUnitSensorEntry 4 }

fcConnUnitSensorMessage

Syntax	SnmpAdminString
Max-Access	read-only
Status	current
Description	This describes the status of the sensor as a message. It may also provide more resolution on the sensor indication, for example 'Cover temperature 1503K, above nominal operating range'
Sequence	::= { fcConnUnitSensorEntry 5 }

fcConnUnitSensorType

Syntax	INTEGER { unknown(1), other(2), battery(3), fan(4), powerSupply(5), transmitter(6), enclosure(7), board(8), receiver(9) }
Max-Access	read-only
Status	current
Description	The type of component being monitored by this sensor. The unknown (1) and other (2) values meanings analogous to those for the fcConnUnitSensorStatus object; all other values are essentially self-explanatory.
Sequence	::= { fcConnUnitSensorEntry 6 }

fcConnUnitSensorCharacteristic

Syntax	INTEGER { unknown(1), other(2), temperature(3), pressure(4), emf(5), currentValue(6), airflow(7), frequency(8), power(9) }
Max-Access	read-only
Status	current
Description	The characteristics being monitored by this sensor. The unknown (1) and other (2) values meanings analogous to those for the fcConnUnitSensorStatus object; emf (5) refers to electro-magnetic field; all other values are essentially self-explanatory.
Sequence	::= { fcConnUnitSensorEntry 7 }
Port Table	The port table contains generic information on ports for a specific connectivity unit.

fcConnUnitPortTable

Syntax	SEQUENCE OF FcConnUnitPortEntry
Max-Access	not-accessible
Status	current
Description	Generic information on ports for a specific connectivity unit.
Sequence	::= { fcMgmtConfig 6 }

fcConnUnitPortEntry

Syntax	FcConnUnitPortEntry
Max-Access	not-accessible
Status	current
Description	Each entry contains the information for a specific port. INDEX { fcConnUnitId, fcConnUnitPortIndex }
Sequence	::= { fcConnUnitPortTable 1 } FcConnUnitPortEntry ::= SEQUENCE { fcConnUnitPortIndex Unsigned32, fcConnUnitPortType INTEGER, fcConnUnitPortFCClassCap FcPortFCClass, fcConnUnitPortFCClassOp FcPortFCClass, fcConnUnitPortState INTEGER, fcConnUnitPortStatus INTEGER, fcConnUnitPortTransmitterType INTEGER, fcConnUnitPortModuleType INTEGER, fcConnUnitPortWwn FcNameId, fcConnUnitPortFCId OCTET STRING, fcConnUnitPortSerialNo SnmpAdminString, fcConnUnitPortRevision SnmpAdminString, fcConnUnitPortVendor SnmpAdminString, fcConnUnitPortSpeed Unsigned32, fcConnUnitPortControl INTEGER, fcConnUnitPortName SnmpAdminString, fcConnUnitPortPhysicalNumber Unsigned32, fcConnUnitPortProtocolCap OCTET STRING, fcConnUnitPortProtocolOp OCTET STRING, fcConnUnitPortNodeWwn FcNameId, fcConnUnitPortHWState INTEGER }

fcConnUnitPortIndex

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	A unique value among all fcConnUnitPortEntry's on this connectivity unit, between 1 and fcConnUnitNumPorts.
Sequence	::= { fcConnUnitPortEntry 1 }

fcConnUnitPortType

Syntax	INTEGER { unknown (1), other (2), notPresent (3), hubPort (4), nPort (5), lPort (6), flPort (7), fPort (8), ePort (9), gPort (10), domainController (11), hubController (12), scsi (13), escon (14), lan (15), wan (16) }
Max-Access	read-only
Status	current

Description The port type refers to the protocol active on the port and can take one of the following values:

unknown (1)	cannot be determined
other (2)	none of the following:
notPresent (3)	no port
hubPort (4)	hub port
nPort (5)	end port for fabric
lPort (6)	end port for loop
flPort (7)	public loop
fPort (8)	fabric port
ePort (9)	fabric expansion port
gPort (10)	generic fabric port
domainController (11)	domain controller
hubController (12)	hub controller
scsi (13)	parallel SCSI port
escon (14)	escon port
lan (15)	LAN port
wan (16)	WAN port"

Sequence ::= { fcConnUnitPortEntry 2 }

fcConnUnitPortFCClassCap

Syntax	FcPortFCClass
Max-Access	read-only
Status	current
Description	Bit mask that specifies the classes of service capability of this port. If this object is not applicable, the agent MUST return all bits set to zero.
Sequence	::= { fcConnUnitPortEntry 3 }

fcConnUnitPortFCClassOp

Syntax	FcPortFCClass
Max-Access	read-only
Status	current

Description Bit mask that specifies the classes of service that are currently operational at this port. If this object is not applicable, the agent MUST return all bits set to zero.

Sequence ::= { fcConnUnitPortEntry 4 }

fcConnUnitPortState

Syntax INTEGER {
 unknown(1),
 online(2),
 offline(3),
 bypassed(4) }

Max-Access read-only

Status current

Description The current state of the port hardware. The bypassed value (4) means that the port is online but is currently being isolated from the loop or fabric for some reason; the other values are essentially self-explanatory. Any value for this object may co-exist with any value for the fcConnUnitPortStatus object.

Sequence ::= { fcConnUnitPortEntry 5 }

fcConnUnitPortStatus

Syntax INTEGER {
 unknown (1),
 unused (2),
 ok (3),
 warning (4),
 failure (5),
 notParticipating (6),
 initializing (7),
 bypassed (8) }

Max-Access read-only

Status	current
Description	The current overall protocol status for the port. The warning value (4) means that the port needs attention; the notParticipating value (6) means that protocol is not being processed; the initializing value (7) means that the port is in the process of coming into service; the bypassed value (8) means that the port has been manually or automatically isolated from the loop or fabric; the other values are essentially self-explanatory. Any value for this object may co-exist with any value for the fcConnUnitPortState object.
Sequence	::= { fcConnUnitPortEntry 6 }

fcConnUnitPortTransmitterType

Syntax	INTEGER { unknown(1), other(2), unused(3), shortwave(4), longwave(5), copper(6), scsi(7), longwaveNoOFC(8), shortwaveNoOFC(9), longwaveLED(10) }
Max-Access	read-only
Status	current
Description	The technology of the port transceiver.
Sequence	::= { fcConnUnitPortEntry 7 }

fcConnUnitPortModuleType

Syntax	INTEGER { unknown(1), other(2),
--------	---------------------------------------

```

        gbic(3),
        embedded(4),
        glm(5),
        gbicSerialId(6),
        gbicNoSerialId(7),
        gbicNotInstalled(8),
        smallFormFactor(9) }

```

Max-Access	read-only
Status	current
Description	The module type of the port connector. This object refers to the hardware implementation of the port. The embedded value (4) means 'fixed' (e.g., oneXnine).
Sequence	::= { fcConnUnitPortEntry 8 }

fcConnUnitPortWwn

Syntax	FcNameId
Max-Access	read-only
Status	current
Description	The World Wide Name of the port if applicable, otherwise empty string.
Sequence	::= { fcConnUnitPortEntry 9 }

fcConnUnitPortFCId

Syntax	OCTET STRING (SIZE(3))
Max-Access	read-only
Status	current
Description	This is the assigned Fibre Channel ID of this port. This value is expected to be a Big Endian value of 24 bits. If this is loop, then it is the ALPA that is connected. If this is an eport, then it will only contain the domain ID left justified, zero filled. If this port does not have a Fibre Channel address, return all bits set to 1.
Sequence	::= { fcConnUnitPortEntry 10 }

fcConnUnitPortSerialNo

Syntax	SnmpAdminString
Max-Access	read-only
Status	current
Description	The serial number identification of the unit (e.g., for a GBIC). If this is not applicable, return a zero-length string.
Sequence	::= { fcConnUnitPortEntry 11 }

fcConnUnitPortRevision

Syntax	SnmpAdminString
Max-Access	read-only
Status	current
Description	The port revision (e.g., for a GBIC).
Sequence	::= { fcConnUnitPortEntry 12 }

fcConnUnitPortVendor

Syntax	SnmpAdminString
Max-Access	read-only
Status	current
Description	The port vendor (e.g., for a GBIC).
Sequence	::= { fcConnUnitPortEntry 13 }

fcConnUnitPortSpeed

Syntax	Unsigned32
UNITS	"kilobytes per second"
Max-Access	read-only
Status	current
Description	The speed of the port in kilobytes per second.
Sequence	::= { fcConnUnitPortEntry 14 }

fcConnUnitPortControl

Syntax	INTEGER { unknown (1), invalid (2), resetConnUnitPort (3), bypassConnUnitPort (4), nbypassConnUnitPort (5), fflineConnUnitPort (6), nlineConnUnitPort (7) }
Max-Access	read-write
Status	current
Description	<p>This object is used to control the addressed fcConnUnit's port. Valid commands are: unknown and invalid are only used as values that are read.</p> <p>resetConnUnitPort (3): If the addressed connectivity unit allows this operation to be performed on this port, the addressed port performs a vendor-specific 'reset' operation. Examples of these operations are: the Link Reset protocol, the Loop Initialization protocol, or a resynchronization occurring between the transceiver in the addressed port to the transceiver that the port is connected to.</p> <p>bypassConnUnitPort (4): If the addressed connectivity unit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'bypass' operation. Examples of these operations are: transitioning from online to offline, a request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.</p> <p>nbypassConnUnitPort (5): If the addressed connectivity unit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'unbypass' operation.</p> <p>Examples of these operations are: the Link Failure protocol, a request (PARTICIPATING) command to the Loop Port state machine, or addition of the port to an arbitrated loop by a hub.</p>

offlineConnUnitPort (6):

If the addressed connectivity unit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'offline' operation.

Examples of these operations are: disabling a port's transceiver, the Link Failure protocol, request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.

onlineConnUnitPort (7):

If the addressed connectivity unit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'online' operation.

Examples of these operations are: enabling a port's transceiver, the Link Failure protocol, request (PARTICIPATING) command to the Loop Port state machine, or addition of the port from an arbitrated loop by a hub.

NOTE: Each implementation may chose not to support SNMP Set operations for any or all of these values. For Sets specifying varbinds for instances of this object and values not supported by a given implementation, the agent will return the SNMP WrongValue PDU error code.

Sequence ::= { fcConnUnitPortEntry 15 }

fcConnUnitPortName

Syntax	SnmpAdminString
Max-Access	read-write
Status	current
Description	A string describing the addressed port.
Sequence	::= { fcConnUnitPortEntry 16 }

fcConnUnitPortPhysicalNumber

Syntax	Unsigned32
Max-Access	read-only
Status	current

Description	This is the internal port number this port is known by. In many implementations, this should be the same as fcConnUnitPortIndex. Some implementations may have an internal port representation not compatible with the rules for table indices. In that case, provide the internal representation of this port in this object. This value may also be used in the fcConnUnitLinkPortNumberX or fcConnUnitLinkPortNumberY objects of the fcConnUnitLinkTable.
Sequence	::= { fcConnUnitPortEntry 17 }

fcConnUnitPortProtocolCap

Syntax	OCTET STRING (SIZE (2))
Max-Access	read-only
Status	current
Description	Bit mask that specifies the driver level protocol capability of this port. If this is not applicable, return all bits set to zero. The bits have the following definition: <ul style="list-style-type: none"> unknown - 0 Loop - 1 Fabric - 2 SCSI - 4 TCP/IP - 8 VI - 16 FICON - 32
Sequence	::= { fcConnUnitPortEntry 18 }

fcConnUnitPortProtocolOp

Syntax	OCTET STRING (SIZE (2))
Max-Access	read-only
Status	current
Description	Bit mask that specifies the driver level protocol(s) that are currently operational. If this is not applicable, return all bits set to zero. This object has the same definition as fcConnUnitPortProtocolCap
Sequence	::= { fcConnUnitPortEntry 19 }

fcConnUnitPortNodeWwn

Syntax	FcNameId
Max-Access	read-only
Status	current
Description	The Node World Wide Name of the port if applicable, otherwise all zeros. This should have the same value for a group of related ports. The container is defined as the largest physical entity. For example, all ports on HBAs on a host will have the same Node WWN. All ports on the same storage subsystem will have the same Node WWN.
Sequence	::= { fcConnUnitPortEntry 20 }

fcConnUnitPortHWState

Syntax	INTEGER { unknown (1), failed (2), bypassed (3), active (4), loopback (5), txfault (6), noMedia (7), linkDown (8) }		
			port failed diagnostics
			FCAL bypass, loop only
			connected to a device
			Port in ext loopback
			Transmitter fault
			media not installed
			waiting for activity (rx sync) }
Max-Access	read-only		
Status	current		
Description	The hardware detected state of the port.		
Sequence	::= { fcConnUnitPortEntry 21 }		

Event group

fcConnUnitEventTable

Syntax	SEQUENCE OF FcConnUnitEventEntry
Max-Access	not-accessible

Status	current
Description	The table of connectivity unit events. Errors, warnings, and information should be reported in this table.
Sequence	::= { fcMgmtConfig 7 }

fcConnUnitEventEntry

Syntax	FcConnUnitEventEntry
Max-Access	not-accessible
Status	current
Description	Each entry contains information on a specific event for the given connectivity unit. INDEX { fcConnUnitId, fcConnUnitEventIndex }
Sequence	::= { fcConnUnitEventTable 1 } FcConnUnitEventEntry ::= SEQUENCE { fcConnUnitEventIndex Unsigned32, fcConnUnitREventTime DisplayString, fcConnUnitSEventTime TimeTicks, fcConnUnitEventSeverity FcEventSeverity, fcConnUnitEventType INTEGER, fcConnUnitEventObject OBJECT IDENTIFIER, fcConnUnitEventDescr SnmpAdminString}

fcConnUnitEventIndex

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	Each connectivity unit has its own event buffer. As it wraps, it may write over previous events. This object is an index into the buffer. It is recommended that this table be read using 'getNext's to retrieve the initial table. The management application should read the event table at periodic intervals and then determine if any new entries were

added by comparing the last known index value with the current highest index value. The management application should then update its copy of the event table. If the read interval is too long, it is possible that there may be events that may not be contained in the agent's internal event buffer.

For example, an agent may read events 50-75. At the next read interval, `fcConnUnitEventCurrID` is 189. If the management app tries to read event index 76, and the agent's internal buffer is 100 entries max, event index 76 will no longer be available. The index value is an incrementing integer starting from one every time there is a table reset. On table reset, all contents are emptied and all indices are set to zero. When an event is added to the table, the event is assigned the next higher integer value than the last item entered into the table. If the index value reaches its maximum value, the next item entered will cause the index value to roll over and start at one again.

Sequence ::= { `fcConnUnitEventEntry 1` }

fcConnUnitREventTime

Syntax	DisplayString (SIZE (15))
Max-Access	read-only
Status	current
Description	This is the real time when the event occurred. It has the following format. DDMMYYYY HHMMSS DD=day number MM=month number YYYY=year number HH=hour number MM=minute number SS=seconds number If not applicable, return a NULL string.
Sequence	::= { <code>fcConnUnitEventEntry 2</code> }

fcConnUnitSevTime

Syntax	TimeTicks
Max-Access	read-only
Status	current
Description	This is the sysuptime timestamp when the event occurred.
Sequence	::= { fcConnUnitEventEntry 3 }

fcConnUnitEventSeverity

Syntax	FcEventSeverity
Max-Access	read-only
Status	current
Description	The event severity level.
Sequence	::= { fcConnUnitEventEntry 4 }

fcConnUnitEventType

Syntax	INTEGER { unknown(1), other(2), status(3), configuration(4), topology(5) }
Max-Access	read-only
Status	current
Description	The type of this event.
Sequence	::= { fcConnUnitEventEntry 5 }

fcConnUnitEventObject

Syntax	OBJECT IDENTIFIER
Max-Access	read-only

Status	current
Description	This is used with the fcConnUnitEventType to identify which object the event refers to. It can be the OID of a connectivity unit or of another object like fcConnUnitPortStatus[...]
Sequence	::= { fcConnUnitEventEntry 6 }

fcConnUnitEventDescr

Syntax	SnmpAdminString
Max-Access	read-only
Status	current
Description	The description of the event.
Sequence	::= { fcConnUnitEventEntry 7 }

The link table is intended to organize and communicate any information the agent possesses which would assist a management application to discover the CONNECTIVITY UNITS in the framework and the TOPOLOGY of their interconnect.

That is, the goal is to assist a management application to both LIST and MAP the elements of the framework. With this goal in mind, the agent SHOULD include as much information as it possesses about any links from its own connectivity units to others, including links among its own units.

An agent SHOULD include partial information about links if it is not able to fully define them in accord with the following structure; however, the information MUST include either a nonzero fcConnUnitNodeId OR a nonzero fcConnUnitPortWwn for each end of the link.

If the agent is able to discover links which do not directly attach to members of its agency and its discovery algorithm gives some assurance the links are recently valid, it MAY include these links.

Link information entered by administrative action MAY be included even if not validated directly if the link has at least one endpoint in this agency, but SHOULD NOT be included otherwise.

A connectivity unit should fill the table in as best it can. One of the methods to fill this in would be to use the RNID ELS (ANSI document 99-422v0). This allows one to query a port for the information needed for the link table.

This table is Max-Accessed either directly if the management software has an index value or via GetNexts. The value of the indexes are not required to be contiguous. Each entry created in this table will be assigned an index. This relationship is kept persistent until the entry is removed from the table or the system is reset. The total number of entries are defined by the size of the table.

For an entry to be considered to be valid, both the X (local) and the Y (remote) need to have one valid value.

fcConnUnitLinkTable

Syntax	SEQUENCE OF FcConnUnitLinkEntry
Max-Access	not-accessible
Status	current
Description	A list of links know to this agent from this connectivity unit to other connectivity units.
Sequence	::= { fcMgmtConfig 8 }

fcConnUnitLinkEntry

Syntax	FcConnUnitLinkEntry
Max-Access	not-accessible
Status	current
Description	An entry describing a particular link to another.
Sequence	INDEX { fcConnUnitId, fcConnUnitLinkIndex }
Sequence	::= { fcConnUnitLinkTable 1 } FcConnUnitLinkEntry ::= SEQUENCE { fcConnUnitLinkIndex Unsigned32, fcConnUnitLinkNodeIdX OCTET STRING, fcConnUnitLinkPortNumberX Integer32, fcConnUnitLinkPortWwnX OCTET STRING, fcConnUnitLinkNodeIdY OCTET STRING, fcConnUnitLinkPortNumberY Integer32,

fcConnUnitLinkPortWwnY	OCTET STRING,
fcConnUnitLinkAgentAddressY	OCTET STRING,
fcConnUnitLinkAgentAddressTypeY	Unsigned32,
fcConnUnitLinkAgentPortY	Unsigned32,
fcConnUnitLinkUnitTypeY	FcUnitType,
fcConnUnitLinkConnIdY	OCTET STRING }

fcConnUnitLinkIndex

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	This value is used to create a unique value for each entry in the link table with the same fcConnUnitId. The value can only be reused if it is not currently in use and the value is the next candidate to be used. This value is allowed to wrap at the highest value represented by the number of bits. This value is reset to zero when the system is reset and the first value to be used is 1.
Sequence	::= { fcConnUnitLinkEntry 1 }

fcConnUnitLinkNodeIdX

Syntax	OCTET STRING (SIZE(64))
Max-Access	read-only
Status	current
Description	The node WWN of the unit at one end of the link. If the node WWN is unknown and the node is an fcConnUnit in the responding agent then the value of this object MUST be equal to its fcConnUnitId.
Sequence	::= { fcConnUnitLinkEntry 2 }

fcConnUnitLinkPortNumberX

Syntax	Integer32
Max-Access	read-only
Status	current

Description The port number on the unit specified by fcConnUnitLinkNodeIdx if known, otherwise -1. If the value is non-negative then it will be equal to fcConnUnitPortPhysicalNumber.

Sequence ::= { fcConnUnitLinkEntry 3 }

fcConnUnitLinkPortWwnX

Syntax OCTET STRING (SIZE(16))

Max-Access read-only

Status current

Description The port WWN of the unit specified by fcConnUnitLinkNodeIdx if known, otherwise 16 octets of binary 0.

Sequence ::= { fcConnUnitLinkEntry 4 }

fcConnUnitLinkNodeIdxY

Syntax OCTET STRING (SIZE(64))

Max-Access read-only

Status current

Description The node WWN of the unit at the other end of the link. If the node WWN is unknown and the node is an fcConnUnit in the responding agent, then the value of this object MUST be equal to its fcConnUnitId.

Sequence ::= { fcConnUnitLinkEntry 5 }

fcConnUnitLinkPortNumberY

Syntax Integer32

Max-Access read-only

Status current

Description The port number on the unit specified by fcConnUnitLinkNodeIdxY if known, otherwise -1. If the value is non-negative then it will be equal to fcConnUnitPortPhysicalNumber.

Sequence ::= { fcConnUnitLinkEntry 6 }

fcConnUnitLinkPortWwnY

Syntax	OCTET STRING (SIZE(16))
Max-Access	read-only
Status	current
Description	The port WWN on the unit specified by fcConnUnitLinkNodeIdY if known, otherwise 16 octets of binary 0.
Sequence	::= { fcConnUnitLinkEntry 7 }

fcConnUnitLinkAgentAddressY

Syntax	OCTET STRING (SIZE(16))
Max-Access	read-only
Status	current
Description	The address of an FCMGMT MIB agent for the node identified by fcConnUnitLinkNodeIdY, if known; otherwise 16 octets of binary 0.
Sequence	::= { fcConnUnitLinkEntry 8 }

fcConnUnitLinkAgentAddressTypeY

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	If fcConnUnitLinkAgentAddressY is non-zero, then it is a protocol address. fcConnUnitLinkAgentAddressTypeY is the the 'address family number' assigned by IANA to identify the address format (eg, 1 is Ipv4, 2 is Ipv6).
Sequence	::= { fcConnUnitLinkEntry 9 }

fcConnUnitLinkAgentPortY

Syntax	Unsigned32
Max-Access	read-only
Status	current

Description The IP port number for the agent. This is provided in case the agent is at a non-standard SNMP port.

Sequence ::= { fcConnUnitLinkEntry 10 }

fcConnUnitLinkUnitTypeY

Syntax FcUnitType

Max-Access read-only

Status current

Description Type of the FC connectivity unit as defined in fcConnUnitType.

Sequence ::= { fcConnUnitLinkEntry 11 }

fcConnUnitLinkConnIdY

Syntax OCTET STRING (SIZE(3))

Max-Access read-only

Status current

Description This is the Fibre Channel ID of this port. If the connectivity unit is a switch, this is expected to be a 24-bit Big Endian value. If this is loop, then it is the ALPA that is connected. If this is an e-port, then it will only contain the domain ID. If not any of those, unknown or cascaded loop, return all bits set to 1.

Sequence ::= { fcConnUnitLinkEntry 12 }

There is one and only one statistics table for each individual port. For all objects in statistics table, if the object is not supported by the conn unit then the high order bit is set to 1 with all other bits set to zero. The high order bit is reserved to indicate if the object is supported or not. All objects start at a value of zero at hardware initialization and continue incrementing till end of 63 bits and then wrap to zero.

Port Statistics

fcConnUnitPortStatTable

Syntax SEQUENCE OF FcConnUnitPortStatEntry

Max-Access not-accessible

Status current

Description A list of statistics for the ports.
 Sequence ::= { fcMgmtStatistics 1 }

fcConnUnitPortStatEntry

Syntax FcConnUnitPortStatEntry
 Max-Access not-accessible
 Status current
 Description An entry describing port statistics.
 INDEX { fcConnUnitId, fcConnUnitPortStatIndex }
 Sequence ::= { fcConnUnitPortStatTable 1 }
 FcConnUnitPortStatEntry ::=
 SEQUENCE {
 fcConnUnitPortStatIndex Unsigned32,
 fcConnUnitPortStatErrs Counter64,
 fcConnUnitPortStatTxObjects Counter64,
 fcConnUnitPortStatRxObjects Counter64,
 fcConnUnitPortStatTxElements Counter64,
 fcConnUnitPortStatRxElements Counter64,
 fcConnUnitPortStatBBCreditZero Counter64,
 fcConnUnitPortStatInputBufsFull Counter64,
 fcConnUnitPortStatFBSYFrames Counter64,
 fcConnUnitPortStatPBSYFrames Counter64,
 fcConnUnitPortStatFRJTFrames Counter64,
 fcConnUnitPortStatPRJTFrames Counter64,
 fcConnUnitPortStatC1RxFrames Counter64,
 fcConnUnitPortStatC1TxFrames Counter64,
 fcConnUnitPortStatC1FBSYFrames Counter64,
 fcConnUnitPortStatC1PBSYFrames Counter64,
 fcConnUnitPortStatC1FRJTFrames Counter64,
 fcConnUnitPortStatC1PRJTFrames Counter64,
 fcConnUnitPortStatC2RxFrames Counter64,

fcConnUnitPortStatC2TxFrames	Counter64,
fcConnUnitPortStatC2FBSYFrames	Counter64,
fcConnUnitPortStatC2PBSYFrames	Counter64,
fcConnUnitPortStatC2FRJTFrames	Counter64,
fcConnUnitPortStatC2PRJTFrames	Counter64,
fcConnUnitPortStatC3RxFrames	Counter64,
fcConnUnitPortStatC3TxFrames	Counter64,
fcConnUnitPortStatC3Discards	Counter64,
fcConnUnitPortStatRxMcastObjects	Counter64,
fcConnUnitPortStatTxMcastObjects	Counter64,
fcConnUnitPortStatRxBcastObjects	Counter64,
fcConnUnitPortStatTxBcastObjects	Counter64,
fcConnUnitPortStatRxLinkResets	Counter64,
fcConnUnitPortStatTxLinkResets	Counter64,
fcConnUnitPortStatLinkResets	Counter64,
fcConnUnitPortStatRxOfflineSeqs	Counter64,
fcConnUnitPortStatTxOfflineSeqs	Counter64,
fcConnUnitPortStatOfflineSeqs	Counter64,
fcConnUnitPortStatLinkFailures	Counter64,
fcConnUnitPortStatInvalidCRC	Counter64,
fcConnUnitPortStatInvalidTxWords	Counter64,
fcConnUnitPortStatPSPErrs	Counter64,
fcConnUnitPortStatLossOfSignal	Counter64,
fcConnUnitPortStatLossOfSync	Counter64,
fcConnUnitPortStatInvOrderedSets	Counter64,
fcConnUnitPortStatFramesTooLong	Counter64,
fcConnUnitPortStatFramesTooShort	Counter64,
fcConnUnitPortStatAddressErrs	Counter64,
fcConnUnitPortStatDelimiterErrs	Counter64,
fcConnUnitPortStatEncodingErrs	Counter64 }

fcConnUnitPortStatIndex

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	A unique value among all entries in this table, between 0 and fcConnUnitNumPort[fcConnUnitPortUnitId].
Sequence	::= { fcConnUnitPortStatEntry 1 }

fcConnUnitPortStatErrs

Syntax	Counter64
Max-Access	read-only
Status	current
Description	A count of the errors that have occurred on this port.
Sequence	::= { fcConnUnitPortStatEntry 2 }

fcConnUnitPortStatTxObjects

Syntax	Counter64
Max-Access	read-only
Status	current
Description	The number of frames/packets/IOs/etc that have been transmitted by this port. Note: A Fibre Channel frame starts with SOF and ends with EOF. FC loop devices should not count frames passed through. This value represents the sum total for all other Tx objects.
Sequence	::= { fcConnUnitPortStatEntry 3 }

fcConnUnitPortStatRxObjects

Syntax	Counter64
Max-Access	read-only
Status	current
Description	The number of frames/packets/IOs/etc that have been received by this port. Note: A Fibre Channel frame starts with SOF and ends with

EOF. FC loop devices should not count frames passed through. This value represents the sum total for all other Rx objects.

Sequence ::= { fcConnUnitPortStatEntry 4 }

fcConnUnitPortStatTxElements

Syntax	Counter64
Max-Access	read-only
Status	current
Description	The number of octets or bytes that have been transmitted by this port. One second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput. Note, for Fibre Channel, ordered sets are not included in the count.
Sequence	::= { fcConnUnitPortStatEntry 5 }

fcConnUnitPortStatRxElements

Syntax	Counter64
Max-Access	read-only
Status	current
Description	The number of octets or bytes that have been received by this port. One second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput. Note, for Fibre Channel, ordered sets are not included in the count.
Sequence	::= { fcConnUnitPortStatEntry 6 }

fcConnUnitPortStatBBCreditZero

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of transitions in/out of BBcredit zero state. The other side is not providing any credit. Note, this is a Fibre Channel stat only.
Sequence	::= { fcConnUnitPortStatEntry 7 }

fcConnUnitPortStatInputBufsFull

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of occurrences when all input buffers of a port were full and outbound buffer-to-buffer credit transitioned to zero. There is no credit to provide to other side. Note, this is a Fibre Channel stat only.
Sequence	::= { fcConnUnitPortStatEntry 8 }

fcConnUnitPortStatFBSYFrames

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of times that FBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. Port can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat. This is the sum of all classes. If you cannot keep the by class counters, then keep the sum counters.
Sequence	::= { fcConnUnitPortStatEntry 9 }

fcConnUnitPortStatPBSYFrames

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of times that PBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if the destination port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat. This is the sum of all classes. If you cannot keep the by class counters, then keep the sum counters.
Sequence	::= { fcConnUnitPortStatEntry 10 }

fcConnUnitPortStatFRJTFrames

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of times that FRJT was returned to this port as a result of a Frame that was rejected by the fabric. Note, this is the total for all classes and is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 11 }

fcConnUnitPortStatPRJTFrames

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of times that FRJT was returned to this port as a result of a Frame that was rejected at the destination N_Port. Note, this is the total for all classes and is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 12 }

fcConnUnitPortStatC1RxFrames

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of Class 1 Frames received at this port. Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 13 }

fcConnUnitPortStatC1TxFrames

Syntax	Counter64
Max-Access	read-only
Status	current

Description Count of Class 1 Frames transmitted out this port. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 14 }

fcConnUnitPortStatC1FBSYFrames

Syntax Counter64

Max-Access read-only

Status current

Description Count of times that FBSY was returned to this port as a result of a Class 1 Frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 15 }

fcConnUnitPortStatC1PBSYFrames

Syntax Counter64

Max-Access read-only

Status current

Description Count of times that PBSY was returned to this port as a result of a Class 1 Frame that could not be delivered to the other end of the link. This occurs if the destination N_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 16 }

fcConnUnitPortStatC1FRJTFrames

Syntax Counter64

Max-Access read-only

Status current

Description Count of times that FRJT was returned to this port as a result of a Class 1 Frame that was rejected by the fabric. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 17 }

fcConnUnitPortStatC1PRJTFrames

Syntax Counter64

Max-Access read-only

Status current

Description Count of times that FRJT was returned to this port as a result of a Class 1 Frame that was rejected at the destination N_Port. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 18 }

fcConnUnitPortStatC2RxFrames

Syntax Counter64

Max-Access read-only

Status current

Description Count of Class 2 Frames received at this port. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 19 }

fcConnUnitPortStatC2TxFrames

Syntax Counter64

Max-Access read-only

Status current

Description Count of Class 2 Frames transmitted out this port. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 20 }

fcConnUnitPortStatC2FBSYFrames

Syntax Counter64

Max-Access read-only

Status current

Description	Count of times that FBSY was returned to this port as a result of a Class 2 Frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 21 }

fcConnUnitPortStatC2PBSYFrames

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of times that PBSY was returned to this port as a result of a Class 2 Frame that could not be delivered to the other end of the link. This occurs if the destination N_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 22 }

fcConnUnitPortStatC2FRJTFrames

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of times that FRJT was returned to this port as a result of a Class 2 Frame that was rejected by the fabric. Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 23 }

fcConnUnitPortStatC2PRJTFrames

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of times that PRJT was returned to this port as a result of a Class 2 Frame that was rejected at the destination N_Port. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 24 }

fcConnUnitPortStatC3RxFrames

Syntax Counter64

Max-Access read-only

Status current

Description Count of Class 3 Frames received at this port. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 25 }

fcConnUnitPortStatC3TxFrames

Syntax Counter64

Max-Access read-only

Status current

Description Count of Class 3 Frames transmitted out of this port. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 26 }

fcConnUnitPortStatC3Discards

Syntax Counter64

Max-Access read-only

Status current

Description Count of Class 3 Frames that were discarded upon reception at this port. There is no FBSY or FRJT generated for Class 3 Frames. They are simply discarded if they cannot be delivered. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 27 }

fcConnUnitPortStatRxMcastObjects

Syntax Counter64

Max-Access read-only

Status	current
Description	Count of Multicast Frames or Packets received at this port.
Sequence	::= { fcConnUnitPortStatEntry 28 }

fcConnUnitPortStatTxMcastObjects

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of Multicast Frames or Packets transmitted out this port.
Sequence	::= { fcConnUnitPortStatEntry 29 }

fcConnUnitPortStatRxBcastObjects

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of Broadcast Frames or Packets received at this port.
Sequence	::= { fcConnUnitPortStatEntry 30 }

fcConnUnitPortStatTxBcastObjects

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of Broadcast Frames or Packets transmitted out this port. On a Fibre Channel loop, count only OPN _r frames generated.
Sequence	::= { fcConnUnitPortStatEntry 31 }

fcConnUnitPortStatRxLinkResets

Syntax	Counter64
Max-Access	read-only
Status	current

Description Count of Link resets. This is the number of LRs received. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 32 }

fcConnUnitPortStatTxLinkResets

Syntax Counter64

Max-Access read-only

Status current

Description Count of Link resets. This is the number LRs transmitted. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 33 }

fcConnUnitPortStatLinkResets

Syntax Counter64

Max-Access read-only

Status current

Description Count of Link resets and LIPs detected at this port. The number times the reset link protocol is initiated. These are the count of the logical resets, a count of the number of primitives. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEvntEntry 34 }

fcConnUnitPortStatRxOfflineSeqs

Syntax Counter64

Max-Access read-only

Status current

Description Count of Offline Primitive OLS received at this port. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 35 }

fcConnUnitPortStatTxOfflineSeqs

Syntax Counter64

Max-Access	read-only
Status	current
Description	Count of Offline Primitive OLS transmitted by this port. Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 36 }

fcConnUnitPortStatOfflineSeqs

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of Offline Primitive sequence received at this port. Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 37 }

fcConnUnitPortStatLinkFailures

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of link failures. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 38 }

fcConnUnitPortStatInvalidCRC

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of frames received with invalid CRC. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Loop ports should not count CRC errors passing through when monitoring. Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 39 }

fcConnUnitPortStatInvalidTxWords

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of invalid transmission words received at this port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 40 }

fcConnUnitPortStatPSPerrs

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of primitive sequence protocol (PSP) errors detected at this port. This count is part of the Link Error Status Block (LESB) FC-PH 29.8). Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 41 }

fcConnUnitPortStatLossOfSignal

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of instances of signal loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 42 }

fcConnUnitPortStatLossOfSync

Syntax	Counter64
Max-Access	read-only
Status	current

Description Count of instances of synchronization loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 43 }

fcConnUnitPortStatInvOrderedSets

Syntax Counter64

Max-Access read-only

Status current

Description Count of invalid ordered sets received at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 44 }

fcConnUnitPortStatFramesTooLong

Syntax Counter64

Max-Access read-only

Status current

Description Count of frames received at this port where the frame length was greater than what was agreed to in FLOGI/PLOGI. This could be caused by losing the end of frame delimiter. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 45 }

fcConnUnitPortStatFramesTooShort

Syntax Counter64

Max-Access read-only

Status current

Description Count of frames received at this port where the frame length was less than the minimum indicated by the frame header - normally 24 bytes, but it could be more if the DFCTL field indicates an optional header should have been present. Note, this is a Fibre Channel only stat.

Sequence ::= { fcConnUnitPortStatEntry 46 }

fcConnUnitPortStatAddressErrs

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of frames received with unknown addressing. e.g. unknown SID or DID. The SID or DID is not known to the routing algorithm. Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 47 }

fcConnUnitPortStatDelimiterErrs

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of invalid frame delimiters received at this port. An example is a frame with a class 2 start and and a class 3 at the end. Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 48 }

fcConnUnitPortStatEncodingErrs

Syntax	Counter64
Max-Access	read-only
Status	current
Description	Count of disparity errors received at this port. Note, this is a Fibre Channel only stat.
Sequence	::= { fcConnUnitPortStatEntry 49 }

Fibre Channel Simple Name Server table

The Fibre Channel Simple Name Server table contains an entry for each device presently known to this fcConnUnit. There will not be any version on this since FC-GS3 does not define a version today.

This table is accessed either directly if the management software has an index value or via GetNexts. The value of the indexes are not required to be contiguous. Each entry created in this table will be

assigned an index. This relationship is kept persistent until the entry is removed from the table or the system is reset. The total number of entries are defined by the size of the table.

fcConnUnitSnsMaxRows

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The maximum number of rows in the fcConnUnitSnsTable table.
Sequence	::= { fcMgmtConfig 9 }

fcConnUnitSnsTable

Syntax	SEQUENCE OF FcConnUnitSnsEntry
Max-Access	not-accessible
Status	current
Description	This table contains an entry for each object registered with this port in the switch.
Sequence	::= { fcMgmtSNS 1 }

fcConnUnitSnsEntry

Syntax	FcConnUnitSnsEntry
Max-Access	not-accessible
Status	current
Description	The Simple Name Server table for the port represented by fcConnUnitSnsPortIndex .
Sequence	INDEX { fcConnUnitId, fcConnUnitSnsPortIndex, fcConnUnitSnsPortIdentifier }
Sequence	::= { fcConnUnitSnsTable 1 } FcConnUnitSnsEntry ::=

```

SEQUENCE {
    fcConnUnitSnsPortIndex          Counter32,
    fcConnUnitSnsPortIdentifier     FcGlobalId, was FcAddressId
                                   (undefined)
    fcConnUnitSnsPortName          FcNameId,
    fcConnUnitSnsNodeName          FcNameId,
    fcConnUnitSnsClassOfSvc        OCTET STRING,
    fcConnUnitSnsNodeIPAddress     OCTET STRING,
    fcConnUnitSnsProcAssoc         OCTET STRING,
    fcConnUnitSnsFC4Type           OCTET STRING,
    fcConnUnitSnsPortType          OCTET STRING,
    fcConnUnitSnsPortIPAddress     OCTET STRING,
    fcConnUnitSnsFabricPortName     FcNameId,
    fcConnUnitSnsHardAddress        FcGlobalId, was FcAddressId
                                   (undefined)
    fcConnUnitSnsSymbolicPortName  DisplayString,
    fcConnUnitSnsSymbolicNodeName  DisplayString }

```

fcConnUnitSnsPortIndex

Syntax	Counter32
Max-Access	read-only
Status	current
Description	The physical port number of this SNS table entry. Each physical port has an SNS table with 1-n entries indexed by ConnUnitSnsPortIdentifier (port address).
Sequence	::= { fcConnUnitSnsEntry 1 }

fcConnUnitSnsPortIdentifier

Syntax	FcGlobalId -- was FcAddressId (undefined)
Max-Access	read-only
Status	current

Description	The Port Identifier for this entry in the SNS table.
Sequence	::= { fcConnUnitSnsEntry 2 }

fcConnUnitSnsPortName

Syntax	FcNameId
Max-Access	read-only
Status	current
Description	The Port WWN for this entry in the SNS table.
Sequence	::= { fcConnUnitSnsEntry 3 }

fcConnUnitSnsNodeName

Syntax	FcNameId
Max-Access	read-only
Status	current
Description	The Node Name for this entry in the SNS table.
Sequence	::= { fcConnUnitSnsEntry 4 }

fcConnUnitSnsClassOfSvc

Syntax	OCTET STRING (SIZE (1))
Max-Access	read-only
Status	current
Description	The Classes of Service offered by this entry in the SNS table.
Sequence	::= { fcConnUnitSnsEntry 5 }

fcConnUnitSnsNodeIPAddress

Syntax	OCTET STRING (SIZE(16))
Max-Access	read-only
Status	current
Description	The IPv6 formatted address of the Node for this entry in the SNS table.

Sequence ::= { fcConnUnitSnsEntry 6 }

fcConnUnitSnsProcAssoc

Syntax OCTET STRING (SIZE (8))

Max-Access read-only

Status current

Description The Process Associator for this entry in the SNS table.

Sequence ::= { fcConnUnitSnsEntry 7 }

fcConnUnitSnsFC4Type

Syntax OCTET STRING (SIZE (32))

Max-Access read-only

Status current

Description The FC-4 Types supported by this entry in the SNS table.

Sequence ::= { fcConnUnitSnsEntry 8 }

fcConnUnitSnsPortType

Syntax OCTET STRING (SIZE (1))

Max-Access read-only

Status current

Description The Port Type of this entry in the SNS table.

Sequence ::= { fcConnUnitSnsEntry 9 }

fcConnUnitSnsPortIPAddress

Syntax OCTET STRING (SIZE(16))

Max-Access read-only

Status current

Description The IPv6 formatted address of this entry in the SNS table.

Sequence ::= { fcConnUnitSnsEntry 10 }

fcConnUnitSnsFabricPortName

Syntax	FcNameId
Max-Access	read-only
Status	current
Description	The Fabric Port name of this entry in the SNS table.
Sequence	::= { fcConnUnitSnsEntry 11 }

fcConnUnitSnsHardAddress

Syntax	FcGlobalId -- was FcAddressId (undefined)
Max-Access	read-only
Status	current
Description	The Hard ALPA of this entry in the SNS table.
Sequence	::= { fcConnUnitSnsEntry 12 }

fcConnUnitSnsSymbolicPortName

Syntax	DisplayString (SIZE (0..79))
Max-Access	read-only
Status	current
Description	The Symbolic Port Name of this entry in the SNS table.
Sequence	::= { fcConnUnitSnsEntry 13 }

fcConnUnitSnsSymbolicNodeName

Syntax	DisplayString (SIZE (0..79))
Max-Access	read-only
Status	current
Description	The Symbolic Node Name of this entry in the SNS table.
Sequence	::= { fcConnUnitSnsEntry 14 }

SNMP trap registration group

fcTrapMaxClients

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	The maximum number of SNMP trap recipients supported by the connectivity unit.
Sequence	::= { fcMgmtNotifyFilter 1 }

fcTrapClientCount

Syntax	Unsigned32
Max-Access	read-only
Status	current
Description	The current number of rows in the trap table.
Sequence	::= { fcMgmtNotifyFilter 2 }

fcTrapRegTable

Syntax	SEQUENCE OF FcTrapRegEntry
Max-Access	not-accessible
Status	current
Description	A table containing a row for each IP address/port number that traps will be sent to.
Sequence	::= { fcMgmtNotifyFilter 3 }

fcTrapRegEntry

Syntax	FcTrapRegEntry
Max-Access	not-accessible
Status	current
Description	Ip/Port pair for a specific client.

INDEX { fcTrapRegIpAddress, fcTrapRegPort }

Sequence ::= { fcTrapRegTable 1 }
 FcTrapRegEntry ::=
 SEQUENCE {
 fcTrapRegIpAddress IpAddress,
 fcTrapRegPort Unsigned32,
 fcTrapRegFilter FcEventSeverity,
 fcTrapRegRowState RowStatus }

fcTrapRegIpAddress

Syntax IpAddress
 Max-Access read-create
 Status current
 Description The Ip address of a client registered for traps.
 Sequence ::= { fcTrapRegEntry 1 }

fcTrapRegPort

Syntax Unsigned32 (1..2147483647)
 Max-Access read-create
 Status current
 Description The UDP port to send traps to for this host. Normally this would be the standard trap port (UDP/162).
 Sequence ::= { fcTrapRegEntry 2 }

fcTrapRegFilter

Syntax FcEventSeverity
 Max-Access read-create
 Status current
 Description This value defines the trap severity filter for this trap host. The fcConnUnit will send to the designated target entity traps that have a severity level less than or equal to this value.

Sequence ::= { fcTrapRegEntry 3 }

fcTrapRegRowState

Syntax	RowStatus
Max-Access	read-create
Status	current
Description	<p>Specifies the operational status of the row. A RowStatus object may take any of six defined values:</p> <p>active: traps may be sent as specified in this row; a management application may change the value of any objects in the row when the status is active.</p> <p>notInService: traps will not be sent using this row.</p> <p>notReady: the conceptual row exists in the agent, but is missing information necessary to send traps (i.e., if any of the other objects in the row are not present or contain invalid values); this value may not be supplied by a management application.</p> <p>createAndGo: supplied by a management application wishing to create a new instance of a conceptual row, supplying valid values for the all the other objects in the row, and have its status automatically set to active, making it available for use in sending traps.</p> <p>createAndWait: supplied by a management application wishing to create a new instance of a conceptual row but not make it available for use in sending traps at that time; and,</p> <p>destroy: supplied by a management application wishing to delete an existing conceptual row.</p>
Sequence	::= { fcTrapRegEntry 4 }

Related traps

fcConnUnitStatusChange

OBJECTS	{ fcConnUnitStatus, fcConnUnitState }
Status	current
Description	The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.
Sequence	::= { fcMgmtNotifications 1 }

fcConnUnitDeletedTrap

OBJECTS	{ fcConnUnitGlobalId }
Status	current
Description	An fcConnUnit has been deleted from this agent. Recommended severity level (for filtering): warning.
Sequence	::= { fcMgmtNotifications 2 }

fcConnUnitEventTrap

OBJECTS	{ fcConnUnitGlobalId, fcConnUnitEventType, fcConnUnitEventObject, fcConnUnitEventDescr }
Status	current
Description	An event has been generated by the connectivity unit. Recommended severity level (for filtering): info.
Sequence	::= { fcMgmtNotifications 3 }

fcConnUnitSensorStatusChange

OBJECTS	{ fcConnUnitSensorStatus }
Status	current

Description	The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.
Sequence	::= { fcMgmtNotifications 4 }

fcConnUnitPortStatusChange

OBJECTS	{ fcConnUnitPortStatus, fcConnUnitPortState }
Status	current
Description	The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.
Sequence	::= { fcMgmtNotifications 5 }

Conformance definitions

(Repeated here from beginning of MIB for ease of reference below.)

fcMgmtNotifications	OBJECT IDENTIFIER	::= { fcMgmtMIB 0 }
fcMgmtObjects	OBJECT IDENTIFIER	::= { fcMgmtMIB 1 }
fcMgmtConformance	OBJECT IDENTIFIER	::= { fcMgmtMIB 2 }
fcMgmtConfig	OBJECT IDENTIFIER	::= { fcMgmtObjects 1 }
fcMgmtNotifyFilter	OBJECT IDENTIFIER	::= { fcMgmtObjects 2 }
fcMgmtStatistics	OBJECT IDENTIFIER	::= { fcMgmtObjects 3 }
fcMgmtSNS	OBJECT IDENTIFIER	::= { fcMgmtObjects 4 }
fcMgmtCompliances	OBJECT IDENTIFIER	::= { fcMgmtConformance 1 }
fcMgmtGroups	OBJECT IDENTIFIER	::= { fcMgmtConformance 2 }

Compliance statements

fcMgmtCompliance

Status	current
Description	<p>The compliance statement for Fibre Channel entities which implement this MIB module.</p> <p>MODULE -- this module</p> <p>MANDATORY-GROUPS {</p> <p>Support for these groups is mandatory for all agents implementing this MIB.</p> <p> fcConnUnitGroup,</p> <p> fcCuEventGroup,</p> <p> fcCuLinkGroup,</p> <p> fcCuPortStatsGroup,</p> <p> fcCuTrapFiltersGroup,</p> <p> fcCuNotificationsGroup }</p>
Group	fcCuSNSGroup
Description	This group is mandatory for agents supporting fibre channel connectivity units that support switch protocol.
Sequence	::= { fcMgmtCompliances 1 }

Conformance units

The fibre channel connectivity unit group

fcConnUnitGroup

Objects {	Scalars
	fcConnUnitNumber,
	fcConnURL,
	fcConnUnitTable
	fcConnUnitGlobalId,

fcConnUnitType,
fcConnUnitNumPorts,
fcConnUnitState,
fcConnUnitStatus,
fcConnUnitProduct,
fcConnUnitSerialNo,
fcConnUnitUpTime,
fcConnUnitUrl,
fcConnUnitDomainId,
fcConnUnitProxyMaster,
fcConnUnitPrincipal,
fcConnUnitNumSensors,
fcConnUnitNumRevs,
fcConnUnitModuleId,
fcConnUnitName,
fcConnUnitInfo,
fcConnUnitControl,
fcConnUnitContact,
fcConnUnitLocation,
fcConnUnitEventFilter,
fcConnUnitNumEvents,
fcConnUnitMaxEvents,
fcConnUnitEventCurrID,

fcConnUnitRevsTable

fcConnUnitRevsRevision,
fcConnUnitRevsDescription,

fcConnUnitSensorTable

fcConnUnitSensorName,
fcConnUnitSensorStatus,
fcConnUnitSensorInfo,

fcConnUnitSensorMessage,
fcConnUnitSensorType,
fcConnUnitSensorCharacteristic,

fcConnUnitPortTable

fcConnUnitPortType,
fcConnUnitPortFCClassCap,
fcConnUnitPortFCClassOp,
fcConnUnitPortState,
fcConnUnitPortStatus,
fcConnUnitPortTransmitterType,
fcConnUnitPortModuleType,
fcConnUnitPortWwn,
fcConnUnitPortFCId,
fcConnUnitPortSerialNo,
fcConnUnitPortRevision,
fcConnUnitPortVendor,
fcConnUnitPortSpeed,
fcConnUnitPortControl,
fcConnUnitPortName,
fcConnUnitPortPhysicalNumber,
fcConnUnitPortProtocolCap,
fcConnUnitPortProtocolOp,
fcConnUnitPortNodeWwn,
fcConnUnitPortHWState}

Status	current
Description	The collection of objects providing Fibre Channel connectivity unit instrumentation and control.
Sequence	::= { fcMgmtGroups 1 }

Event group

fcCuEventGroup

Objects	{ fcConnUnitEventTable fcConnUnitEventIndex, fcConnUnitREventTime, fcConnUnitSEventTime, fcConnUnitEventSeverity, fcConnUnitEventType, fcConnUnitEventObject, fcConnUnitEventDescr }
Status	current
Description	The collection of objects providing Fibre Channel connectivity unit event information.
Sequence	::= { fcMgmtGroups 2 }

Link group

fcCuLinkGroup

Objects	{ fcConnUnitLinkTable fcConnUnitLinkIndex, cConnUnitLinkNodeIdX, cConnUnitLinkPortNumberX, cConnUnitLinkPortWwnX, cConnUnitLinkNodeIdY, cConnUnitLinkPortNumberY, cConnUnitLinkPortWwnY, cConnUnitLinkAgentAddressY, cConnUnitLinkAgentAddressTypeY, cConnUnitLinkAgentPortY,
---------	---

```

cConnUnitLinkUnitTypeY,
cConnUnitLinkConnIdY }

```

Status	current
Description	The collection of objects providing Fibre Channel connectivity unit link (topology) information.
Sequence	::= { fcMgmtGroups 3 }

Port statistics group

fcCuPortStatsGroup

Objects	<pre> {fcConnUnitPortStatTable fcConnUnitPortStatIndex, fconnUnitPortStatErrs, fcConnUnitPortStatTxObjects, fcConnUnitPortStatRxObjects, fcConnUnitPortStatTxElements, fcConnUnitPortStatRxElements, fcConnUnitPortStatBBCreditZero, fcConnUnitPortStatInputBufsFull, fcConnUnitPortStatFBSYFrames, fcConnUnitPortStatPBSYFrames, fcConnUnitPortStatFRJTFrames, fcConnUnitPortStatPRJTFrames, fcConnUnitPortStatC1RxFrames, fcConnUnitPortStatC1TxFrames, fcConnUnitPortStatC1FBSYFrames, fcConnUnitPortStatC1PBSYFrames, fcConnUnitPortStatC1FRJTFrames, fcConnUnitPortStatC1PRJTFrames, fcConnUnitPortStatC2RxFrames, </pre>
---------	--

fcConnUnitPortStatC2TxFrames,
fcConnUnitPortStatC2FBSYFrames,
fcConnUnitPortStatC2PBSYFrames,
fcConnUnitPortStatC2FRJTFrames,
fcConnUnitPortStatC2PRJTFrames,
fcConnUnitPortStatC3RxFrames,
fcConnUnitPortStatC3TxFrames,
fcConnUnitPortStatC3Discards,
fcConnUnitPortStatRxMcastObjects,
fcConnUnitPortStatTxMcastObjects,
fcConnUnitPortStatRxBcastObjects,
fcConnUnitPortStatTxBcastObjects,
fcConnUnitPortStatRxLinkResets,
fcConnUnitPortStatTxLinkResets,
fcConnUnitPortStatLinkResets,
fcConnUnitPortStatRxOfflineSeqs,
fcConnUnitPortStatTxOfflineSeqs,
fcConnUnitPortStatOfflineSeqs,
fcConnUnitPortStatLinkFailures,
fcConnUnitPortStatInvalidCRC,
fcConnUnitPortStatInvalidTxWords,
fcConnUnitPortStatPSPErrs,
fcConnUnitPortStatLossOfSignal,
fcConnUnitPortStatLossOfSync,
fcConnUnitPortStatInvOrderedSets,
fcConnUnitPortStatFramesTooLong,
fcConnUnitPortStatFramesTooShort,
fcConnUnitPortStatAddressErrs,
fcConnUnitPortStatDelimiterErrs,

fcConnUnitPortStatEncodingErrs }

Status	current
Description	The collection of objects providing Fibre Channel connectivity unit port statistics.
Sequence	::= { fcMgmtGroups 4 }

Fibre Channel Simple Name Server group

fcCuSNSGroup

Objects	{
Scalars	fcConnUnitSnsMaxRows,
fcConnUnitSnsTable	fcConnUnitSnsPortIndex,
	fcConnUnitSnsPortIdentifier,
	fcConnUnitSnsPortName,
	fcConnUnitSnsNodeName,
	fcConnUnitSnsClassOfSvc,
	fcConnUnitSnsNodeIPAddress,
	fcConnUnitSnsProcAssoc,
	fcConnUnitSnsFC4Type,
	fcConnUnitSnsPortType,
	fcConnUnitSnsPortIPAddress,
	fcConnUnitSnsFabricPortName,
	fcConnUnitSnsHardAddress,
	fcConnUnitSnsSymbolicPortName,
	fcConnUnitSnsSymbolicNodeName }
Status	current
Description	The collection of objects providing Fibre Channel connectivity unit simple name server information.
Sequence	::= { fcMgmtGroups 5 }

SNMP trap filter group

fcCuTrapFiltersGroup

Objects	{ Scalars fcTrapMaxClients, fcTrapClientCount, fcTrapRegTable fcTrapRegIpAddress, fcTrapRegPort, fcTrapRegFilter, fcTrapRegRowState }
Status	current
Description	The collection of objects controlling SNMP notification (i.e., trap) destinations.
Sequence	::= { fcMgmtGroups 6 }

FC-MGMT-MIB
notifications group

fcCuNotificationsGroup

Notifications	{fcConnUnitStatusChange, fcConnUnitDeletedTrap, fcConnUnitEventTrap, fcConnUnitSensorStatusChange, fcConnUnitPortStatusChange }
Status	current
Description	The set of SNMP notifications which an agent is required to implement.
Sequence	::= { fcMgmtGroups 7 } END

FCMGMT-MIB Definitions

Version 3.0 of FA MIB

```
IMPORTS
    IpAddress, TimeTicks, experimental
        FROM RFC1155-SMI
    OBJECT-Type
        FROM RFC-1212
    DisplayString
        FROM RFC1213-MIB
    TRAP-Type
        FROM RFC-1215;
```

Textual conventions for this MIB

```
FcNameId ::= OCTET STRING (SIZE(8))
FcGlobalId ::= OCTET STRING (SIZE(16))
FcAddressId ::= OCTET STRING (SIZE(3))
FcEventSeverity ::= INTEGER {
    unknown (1),
    emergency (2),
    alert (3),
```

```

critical (4),
error (5),
warning (6),
notify (7),
info (8),
debug (9),
mark (10) -- All messages logged
}

```

```

FcUnitType ::= INTEGER {
unknown(1),
other(2),                none of the following
hub(3),                  passive connectivity unit supporting loop
                           protocol.
switch(4),               active connectivity unit supporting
                           multiple protocols.
gateway(5),              unit that converts not only the interface
                           but also encapsulates the frame into
                           another protocol. The assumption is that
                           there is always two gateways connected
                           together. For example, FC <-> ATM.
converter(6),            unit that converts from one interface to
                           another. For example, FC <-> SCSI.
hba(7),                  host bus adapter
proxy-agent(8),          software proxy-agent
storage-device(9),       disk, cd, tape, etc.
host(10),                host computer
storage-subsystem(11),   raid, library, etc.
module(12),              subcomponent of a system
swdriver(13),            software driver

```

storage-access-device(14), Provides storage management and access
for heterogeneous hosts and
heterogeneous devices.

wdm(15), waveform division multiplexer
ups(16) uninterruptable power supply }

fcmgmt OBJECT IDENTIFIER ::= { experimental 94 }

Groups in fcmgmt

connSet	OBJECT IDENTIFIER	::= { fcmgmt 1 }
trapReg	OBJECT IDENTIFIER	::= { fcmgmt 2 }
statSet	OBJECT IDENTIFIER	::= { fcmgmt 4 }
connUnitServiceSet	OBJECT IDENTIFIER	::= { fcmgmt 5 }
connUnitServiceScalars	OBJECT IDENTIFIER	::={ connUnitServiceSet 1 }
connUnitServiceTables	OBJECT IDENTIFIER	::={ connUnitServiceSet 2 }

revisionNumber

Syntax	DisplayString (SIZE (4))
Access	read-only
Status	mandatory
Description	<p>This is the revision number for this MIB. The format of the revision value is as follows</p> <p>(0) = high order major revision number</p> <p>(1) = low order major revision number</p> <p>(2) = high order minor revision number</p> <p>(3) = low order minor revision number</p> <p>The value will be stored as an ASCII value. The following is the current value of this object.</p>

- (0) = '0'
- (1) = '3'
- (2) = '0'
- (3) = '0'

This defines a revision of 03.00

Sequence ::= { fcmgmt 3 }

Connectivity unit group

Implementation of the group is mandatory for all systems.

uNumber

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The number of connectivity units present on this system (represented by this agent). May be a count of the boards in a chassis or the number of full boxes in a rack. DEFVAL { 1 }
Sequence	::= { connSet 1 }

systemURL

Syntax	DisplayString
Access	read-write
Status	mandatory
Description	The top-level URL of the system. If it does not exist the value is empty string. The URL format is implementation dependant and can have keywords embedded that are preceded by a percent sign (e.g., %USER).

The following are the defined keywords that will be recognized and replaced with data during a launch.

USER	replace with username
PASSWORD	replace with password
GLOBALID	replace with globalid
SERIALNO	replace with serial number

If write is not supported, then return invalid. This value will be retained across boots.

DEFVAL { "" }

Sequence ::= { connSet 2 }

statusChangeTime

Syntax	TimeTicks
Access	read-only
Status	obsolete
Description	The sysuptime timestamp in centiseconds at which the last status change occurred for any members of the set.
Sequence	::= { connSet 3 }

configurationChangeTime

Syntax	TimeTicks
Access	read-only
Status	obsolete
Description	The sysuptime timestamp in centiseconds at which the last configuration change occurred for any members of the set. This represents a union of change information for connUnitConfigurationChangeTime.
Sequence	::= { connSet 4 }

connUnitTableChangeTime

Syntax	TimeTicks
Access	read-only

Status	obsolete
Description	The sysuptime timestamp in centiseconds at which the connUnitTable was updated (an entry was either added or deleted).
Sequence	::= { connSet 5 }

Connectivity Table

The Connectivity table contains general information on the system's units.

connUnitTable

Syntax	SEQUENCE OF ConnUnitEntry
Access	not-accessible
Status	mandatory
Description	A list of units under a single SNMP agent. The number of entries is given by the value of uNumber. It is 1 for stand-alone system.
Sequence	::= { connSet 6 }

connUnitEntry

Syntax	ConnUnitEntry
Access	not-accessible
Status	mandatory
Description	A connectivity unit entry containing objects for a particular unit. INDEX { connUnitId }
Sequence	::= { connUnitTable 1 } ConnUnitEntry ::= SEQUENCE { connUnitId FcGlobalId, connUnitGlobalId FcGlobalId, connUnitType

FcUnitType,
connUnitNumports
INTEGER,
connUnitState
INTEGER,
connUnitStatus
INTEGER,
connUnitProduct
DisplayString,
connUnitSn
DisplayString,
connUnitUpTime
TimeTicks,
connUnitUrl
DisplayString,
connUnitDomainId
OCTET STRING,
connUnitProxyMaster
INTEGER,
connUnitPrincipal
INTEGER,
connUnitNumSensors
INTEGER,
connUnitStatusChangeTime
TimeTicks,
connUnitConfigurationChangeTime
TimeTicks,
connUnitNumRevs
INTEGER,

```
connUnitNumZones
    INTEGER,
connUnitModuleId
    FcGlobalId,
connUnitName
    DisplayString,
connUnitInfo
    DisplayString,
connUnitControl
    INTEGER,
connUnitContact
    DisplayString,
connUnitLocation
    DisplayString,
connUnitEventFilter
    FcEventSeverity,
connUnitNumEvents
    INTEGER,
connUnitMaxEvents
    INTEGER,
connUnitEventCurrID
    INTEGER }
```

connUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The unique identification for this connectivity unit among those within this proxy domain. The value MUST be unique within the proxy domain because it is the index variable for connUnitTable.

The value assigned to a given connectivity unit SHOULD be persistent across agent and unit resets. It SHOULD be the same as connUnitGlobalId if connUnitGlobalId is known and stable.

Sequence ::= { connUnitEntry 1 }

connUnitGlobalId

Syntax FcGlobalId
Access read-only
Status mandatory

Description An optional global-scope identifier for this connectivity unit. It MUST be a WWN for this connectivity unit or 16 octets of value zero. WWN formats requiring fewer than 16 octets MUST be extended to 16 octets with trailing zero octets, Left justified, zero filled, If a WWN is used for connUnitId, the same WWN MUST be used forconnUnitGlobalId.

When a non-zero value is provided, it SHOULD be persistent across agent and unit resets. It SHOULD be globally unique. It SHOULD be one of these FC-PH/PH3 formats:

- IEEE (NAA=1)
- IEEE Extended (NAA=2)
- IEEE Registered (NAA=5).
- IEEE Registered extended (NAA=6).

Use of the IEEE formats allows any IEEE-registered vendor to assure global uniqueness independently. The following are some references on IEEE WWN formats:

<http://standards.ieee.org/regauth/oui/tutorials/fibreformat.html>
[http://standards.ieee.org/regauth/oui/tutorials/fibrecomp_id.htm](http://standards.ieee.org/regauth/oui/tutorials/fibrecomp_id.html)
l

If one or more WWNs are associated with the connUnit via other management methods, one of them SHOULD be used for connUnitGlobalId.

If there is not a WWN assigned specifically to the connUnit, there is some merit, though not a requirement, to using a WWN assigned to (one of) its permanently attached FC/LAN interface(s). This can not risk uniqueness, though.

As a counterexample, if your agent runs in a host and the host has an HBA, it is quite possible that agent, host, and HBA will all be distinct connUnits, so the host and agent can not use the WWN of the HBA.

Another example:

If your hub has a built-in Ethernet port, it might be reasonable for the hub to use its MAC address (prefixed with the appropriate NAA) as its connUnitId. But if the Ethernet were a replaceable PCCard, the hub should have an independent ID.

Sequence ::= { connUnitEntry 2 }

connUnitType

Syntax	FcUnitType
Access	read-only
Status	mandatory
Description	The type of this connectivity unit.
Sequence	::= { connUnitEntry 3 }

connUnitNumports

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Number of physical ports in the connectivity unit (internal/ embedded, external).
Sequence	::= { connUnitEntry 4 }

connUnitState

Syntax	INTEGER { unknown(1), online(2), available for meaningful work offline(3) unavailable for meaningful work, for example in self-test mode, configuration, etc. }
--------	--

Access	read-only
Status	mandatory
Description	Overall state of the connectivity unit.
Sequence	::= { connUnitEntry 5 }

connUnitStatus

Syntax	INTEGER { unknown(1), unused(2), cannot report status ok(3), available for meaningful work warning(4), something needs attention failed(5) something has failed } }
Access	read-only
Status	mandatory
Description	Overall status of the connectivity unit. The goal of this object is to be the single poll point to check the status of the connunit. If there is any other component that has warning, then this should be set to warning, etc.
Sequence	::= { connUnitEntry 6 }

connUnitProduct

Syntax	DisplayString (SIZE (0..79))
Access	read-only
Status	mandatory
Description	The connectivity unit vendor's product model name.
Sequence	::= { connUnitEntry 7 }

connUnitSn

Syntax	DisplayString (SIZE (0..79))
Access	read-only

Status	mandatory
Description	The serial number for this connectivity unit.
Sequence	::= { connUnitEntry 8 }

connUnitUpTime

Syntax	TimeTicks
Access	read-only
Status	mandatory
Description	The number of centiseconds since the last unit initialization.
Sequence	::= { connUnitEntry 9 }

connUnitUrl

Syntax	DisplayString
Access	read-write
Status	mandatory
Description	URL to launch a management application, if applicable. Otherwise empty string. In a standalone unit, this would be the same as the top-level URL. This has the same definition as systemURL for keywords. If write is not supported, then return invalid. This value will be retained across boots.
Sequence	::= { connUnitEntry 10 }

connUnitDomainId

Syntax	OCTET STRING (SIZE(3))
Access	read-only
Status	mandatory
Description	24 bit Fibre Channel address ID of this connectivity unit, right justified with leading zero's if required. This should be set to the Fibre Channel address ID or if it is a switch it would be set to the Domain Controller address. If this value is not applicable, return all bits set to one.
Sequence	::= { connUnitEntry 11 }

connUnitProxyMaster

Syntax	INTEGER { unknown(1), no(2), yes(3) }
Access	read-only
Status	mandatory
Description	A value of 'yes' means this is the proxy master unit for a set of managed units. For example, this could be the only unit with a management card in it for a set of units. A standalone unit should return 'yes' for this object.
Sequence	::= { connUnitEntry 12 }

connUnitPrincipal

Syntax	INTEGER {unknown(1), no(2), yes(3) }
Access	read-only
Status	mandatory
Description	Whether this connectivity unit is the principal unit within the group of fabric elements. If this value is not applicable, return unknown.
Sequence	::= { connUnitEntry 13 }

connUnitNumSensors

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Number of sensors in the connUnitSensorTable.
Sequence	::= { connUnitEntry 14 }

connUnitStatusChangeTime

Syntax	TimeTicks
Access	read-only
Status	obsolete
Description	The sysuptime timestamp in centiseconds at which the last status change occurred.

Sequence ::= { connUnitEntry 15 }

connUnitConfigurationChangeTime

Syntax TimeTicks
 Access read-only
 Status obsolete
 Description The sysuptime timestamp in centiseconds at which the last configuration change occurred.
 Sequence ::= { connUnitEntry 16 }

connUnitNumRevs

Syntax INTEGER
 Access read-only
 Status mandatory
 Description The number of revisions in the connUnitRevsTable.
 DEFVAL { 1 }
 Sequence ::= { connUnitEntry 17 }

connUnitNumZones

Syntax INTEGER
 Access read-only
 Status obsolete
 Description Number of zones defined in connUnitZoneTable.
 Sequence ::= { connUnitEntry 18 }

connUnitModuleId

Syntax FcGlobalId
 Access read-only
 Status mandatory

Description This is a unique id, persistent between boots, that can be used to group a set of connUnits together into a module. The intended use would be to create a connUnit with a connUnitType of 'module' to represent a physical or logical group of connectivity units. Then the value of the group would be set to the value of connUnitId for this 'container' connUnit. connUnitModuleId should be zeros if this connUnit is not part of a module.

Sequence ::= { connUnitEntry 19 }

connUnitName

Syntax DisplayString (SIZE(0..79))

Access read-write

Status mandatory

Description A display string containing a name for this connectivity unit. This object value should be persistent between boots.

Sequence ::= { connUnitEntry 20 }

connUnitInfo

Syntax DisplayString

Access read-write

Status mandatory

Description A display string containing information about this connectivity unit. This object value should be persistent between boots.

Sequence ::= { connUnitEntry 21 }

connUnitControl

Syntax INTEGER {
 unknown(1),
 invalid(2),
 resetConnUnitColdStart(3),
 resetConnUnitWarmStart(4),
 offlineConnUnit(5),

onlineConnUnit(6) }

Access	read-write
Status	mandatory
Description	This object is used to control the addressed connUnit.

NOTE: 'Cold Start' and 'Warm Start' are as defined in MIB II and are not meant to be a factory reset.

resetConnUnitColdStart	the addressed unit performs a 'Cold Start' reset.
resetConnUnitWarmStart	the addressed unit performs a 'Warm Start' reset.
offlineConnUnit	the addressed unit puts itself into an implementation dependant 'offline' state. In general, if a unit is in an offline state, it cannot be used to perform meaningful Fibre Channel work.
onlineConnUnit	the addressed unit puts itself into an implementation dependant 'online' state. In general, if a unit is in an online state, it is capable of performing meaningful Fibre Channel work.

NOTE: Each implementation may chose not to allow any or all of these values on a SET.

Sequence ::= { connUnitEntry 22 }

connUnitContact

Syntax	DisplayString (SIZE (0..79))
Access	read-write
Status	mandatory
Description	Contact information for this connectivity unit. Persistent across boots.
Sequence	::= { connUnitEntry 23 }

connUnitLocation

Syntax	DisplayString (SIZE (0..79))
Access	read-write
Status	mandatory
Description	Location information for this connectivity unit. Persistent across boots.
Sequence	::= { connUnitEntry 24 }

connUnitEventFilter

Syntax	FcEventSeverity
Access	read-write
Status	mandatory
Description	This value defines the event severity that will be logged by this connectivity unit. All events of severity less than or equal to connUnitEventFilter are logged in connUnitEventTable. Persistent across boots.
Sequence	::= { connUnitEntry 25 }

connUnitNumEvents

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Number of events currently in the connUnitEventTable.
Sequence	::= { connUnitEntry 26 }

connUnitMaxEvents

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Max number of events that can be defined in connUnitEventTable.

Sequence ::= { connUnitEntry 27 }

connUnitEventCurrID

Syntax INTEGER
 Access read-only
 Status mandatory
 Description The last used event id (connUnitEventIndex).
 Sequence ::= { connUnitEntry 28 }

The Table of revisions for hardware and software elements.

connUnitRevsTable

Syntax SEQUENCE OF ConnUnitRevsEntry
 Access not-accessible
 Status mandatory
 Description Table of the revisions supported by connectivity units managed by this agent.
 Sequence ::= { connSet 7 }

connUnitRevsEntry

Syntax ConnUnitRevsEntry
 Access not-accessible
 Status mandatory
 Description ""
 INDEX { connUnitRevsUnitId, connUnitRevsIndex }
 Sequence ::= { connUnitRevsTable 1 }
 ConnUnitRevsEntry ::=
 SEQUENCE {
 connUnitRevsUnitId
 FcGlobalId,

```

connUnitRevsIndex
    INTEGER,
connUnitRevsRevId
    DisplayString,
connUnitRevsDescription
    DisplayString }

```

connUnitRevsUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this revision table.
Sequence	::= { connUnitRevsEntry 1 }

connUnitRevsIndex

Syntax	INTEGER (1..2147483647)
Access	read-only
Status	mandatory
Description	A unique value among all connUnitRevsEntrys with the same value of connUnitRevsUnitId, in the range between 1 and connUnitNumRevs[connUnitRevsUnitId].
Sequence	::= { connUnitRevsEntry 2 }

connUnitRevsRevId

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	A vendor-specific string identifying a revision of a component of the connUnit indexed by connUnitRevsUnitId.
Sequence	::= { connUnitRevsEntry 3 }

connUnitRevsDescription

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	Description of a component to which the revision corresponds.
Sequence	::= { connUnitRevsEntry 4 }

Sensor table

connUnitSensorTable

Syntax	SEQUENCE OF ConnUnitSensorEntry
Access	not-accessible
Status	mandatory
Description	Table of the sensors supported by each connectivity unit managed by this agent.
Sequence	::= { connSet 8 }

connUnitSensorEntry

Syntax	ConnUnitSensorEntry
Access	not-accessible
Status	mandatory
Description	Each entry contains the information for a specific sensor. INDEX { connUnitSensorUnitId, connUnitSensorIndex }
Sequence	::= { connUnitSensorTable 1 } ConnUnitSensorEntry ::= SEQUENCE { connUnitSensorUnitId FcGlobalId, connUnitSensorIndex


```

        INTEGER (1..2147483647),
connUnitSensorName
        DisplayString,
connUnitSensorStatus
        INTEGER,
connUnitSensorInfo
        DisplayString,
connUnitSensorMessage
        DisplayString,
connUnitSensorType
        INTEGER,
connUnitSensorCharacteristic
        INTEGER }

```

connUnitSensorUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this sensor table.
Sequence	::= { connUnitSensorEntry 1 }

connUnitSensorIndex

Syntax	INTEGER (1..2147483647)
Access	read-only
Status	mandatory
Description	A unique value among all connUnitSensorEntryS with the same value of connUnitSensorUnitId, in the range between 1 and connUnitNumSensor[connUnitSensorUnitId].
Sequence	::= { connUnitSensorEntry 2 }

connUnitSensorName

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	A textual identification of the sensor intended primarily for operator use.
Sequence	::= { connUnitSensorEntry 3 }

connUnitSensorStatus

Syntax	INTEGER { unknown(1), other(2), the sensor indicates other than ok, warning or failure. ok(3), the sensor indicates ok warning(4), the sensor indicates a warning failed(5) the sensor indicates failure }
Access	read-only
Status	mandatory
Description	The status indicated by the sensor.
Sequence	::= { connUnitSensorEntry 4 }

connUnitSensorInfo

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	Miscellaneous static info about the sensor such as its serial number.
Sequence	::= { connUnitSensorEntry 5 }

connUnitSensorMessage

Syntax	DisplayString
--------	---------------

Access	read-only
Status	mandatory
Description	This describes the status of the sensor as a message. It may also provide more resolution on the sensor indication, for example 'Cover temperature 1503K, above nominal operating range'
Sequence	::= { connUnitSensorEntry 6 }

connUnitSensorType

Syntax	INTEGER { unknown(1), other(2), battery(3), fan(4), power-supply(5), transmitter(6), enclosure(7), board(8), receiver(9) }
Access	read-only
Status	mandatory
Description	The type of component being monitored by this sensor.
Sequence	::= { connUnitSensorEntry 7 }

connUnitSensorCharacteristic

Syntax	INTEGER { unknown(1), other(2), temperature(3), pressure(4), emf(5),
--------	---

	currentValue(6), -- current is a keyword airflow(7), frequency(8), power(9), door(10) }
Access	read-only
Status	mandatory
Description	The characteristics being monitored by this sensor.
Sequence	::= { connUnitSensorEntry 8 }

Port Table

connUnitPortTable

Syntax	SEQUENCE OF ConnUnitPortEntry
Access	not-accessible
Status	mandatory
Description	Generic information on ports for a specific connUnit.
Sequence	::= { connSet 10 }

connUnitPortEntry

Syntax	ConnUnitPortEntry
Access	not-accessible
Status	mandatory
Description	Each entry contains the information for a specific port. INDEX { connUnitPortUnitId, connUnitPortIndex }
Sequence	::= { connUnitPortTable 1 } ConnUnitPortEntry ::=

```

SEQUENCE {
    connUnitPortUnitId          FcGlobalId,
    connUnitPortIndex          INTEGER,
    connUnitPortType            INTEGER,
    connUnitPortFcClassCap     OCTET STRING,
    connUnitPortFcClassOp      OCTET STRING,
    connUnitPortState          INTEGER,
    connUnitPortStatus          INTEGER,
    connUnitPortTransmitterType INTEGER,
    connUnitPortModuleType     INTEGER,
    connUnitPortWwn            FcNameId,
    connUnitPortFCId           FcAddressId,
    connUnitPortSn             DisplayString,
    connUnitPortRevision       DisplayString,
    connUnitPortVendor         DisplayString,
    connUnitPortSpeed          INTEGER,
    connUnitPortControl        INTEGER,
    connUnitPortName           DisplayString,
    connUnitPortPhysicalNumber INTEGER,
    connUnitPortStatObject     OBJECT IDENTIFIER,
    connUnitPortProtocolCap    OCTET STRING,
    connUnitPortProtocolOp     OCTET STRING,
    connUnitPortNodeWwn        FcNameId,
    connUnitPortHWState        INTEGER }

```

connUnitPortUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this port.
Sequence	::= { connUnitPortEntry 1 }

connUnitPortIndex

Syntax	INTEGER (1..2147483647)
Access	read-only
Status	mandatory
Description	A unique value among all connUnitPortEntries on this connectivity unit, between 1 and connUnitNumPort[connUnitPortUnitId].
Sequence	::= { connUnitPortEntry 2 }

connUnitPortType

Syntax	INTEGER { unknown (1), other (2), not-present (3), hub-port (4), n-port (5), end port for fabric nl-port (6), end port for loop fl-port (7), public loop f-port (8), fabric port e-port (9), fabric expansion port g-port (10), generic fabric port domain-ctl (11), domain controller hub-controller(12), scsi (13), parallel SCSI port escon (14), lan (15), wan (16), ac (17), AC power line dc (18), DC power line ssa (19) serial storage architecture }
Access	read-only
Status	mandatory

Description The port type.

Sequence ::= { connUnitPortEntry 3 }

connUnitPortFCClassCap

Syntax OCTET STRING (SIZE (2))

Access read-only

Status mandatory

Description Bit mask that specifies the classes of service capability of this port. If this is not applicable, return all bits set to zero.

 The bits have the following definition:

unknown	0
class-f	1
class-one	2
class-two	4
class-three	8
class-four	16
class-five	32
class-six	64

Sequence ::= { connUnitPortEntry 4 }

connUnitPortFCClassOp

Syntax OCTET STRING (SIZE (2))

Access read-only

Status mandatory

Description Bit mask that specifies the classes of service that are currently operational. If this is not applicable, return all bits set to zero. This object has the same definition as connUnitPortFCClassCap.

Sequence ::= { connUnitPortEntry 5 }

connUnitPortState

Syntax	INTEGER { unknown(1), online(2), available for meaningful work offline(3), not available for meaningful work bypassed(4), no longer used (4/12/00) diagnostics(5) }
Access	read-only
Status	mandatory
Description	The user selected state of the port hardware.
Sequence	::= { connUnitPortEntry 6 }

connUnitPortStatus

Syntax	INTEGER { unknown (1), unused (2), device cannot report this status ready (3), FCAL Loop or FCPH Link reset protocol initialization has completed warning (4), do not use (4/12/00) failure (5), do not use (4/12/00) notparticipating (6), loop notparticipating and does not have a loop address initializing (7), protocol is proceeding bypass (8), do not use (4/12/00) ols (9) FCP offline status }
Access	read-only
Status	mandatory
Description	An overall protocol status for the port. This value of connUnitPortState is not online, then this is reported Unknown.
Sequence	::= { connUnitPortEntry 7 }

connUnitPortTransmitterType

Syntax	INTEGER { unknown(1), other(2), unused(3), shortwave(4), longwave(5), copper(6), scsi(7), longwaveNoOFC(8), shortwaveNoOFC(9), longwaveLED(10), ssa(11) }
Access	read-only
Status	mandatory
Description	The technology of the port transceiver.
Sequence	::= { connUnitPortEntry 8 }

connUnitPortModuleType

Syntax	INTEGER { unknown(1), other(2), gbic(3), embedded(4), -- fixed, i.e., oneXnine glm(5), gbicSerialId(6), gbicNoSerialId(7), gbicNotInstalled(8),
--------	---

smallFormFactor(9) -- this is generically a small form factor connector.

}

Access read-only
 Status mandatory
 Description The module type of the port connector.
 Sequence ::= { connUnitPortEntry 9 }

connUnitPortWwn

Syntax FcNameId
 Access read-only
 Status mandatory
 Description The World Wide Name of the port if applicable, otherwise all zeros.
 Sequence ::= { connUnitPortEntry 10 }

connUnitPortFCId

Syntax FcAddressId
 Access read-only
 Status mandatory
 Description This is the assigned Fibre Channel ID of this port. This value is expected to be a Big Endian value of 24 bits. If this is loop, then it is the ALPA that is connected.
 If this is an eport, then it will only contain the domain ID left justified, zero filled. If this port does not have a Fibre Channel address, return all bits set to 1.
 Sequence ::= { connUnitPortEntry 11 }

connUnitPortSn

Syntax DisplayString (SIZE(0..79))
 Access read-only
 Status mandatory

Description The serial number of the unit (e.g., for a GBIC). If this is not applicable, return empty string.

Sequence ::= { connUnitPortEntry 12 }

connUnitPortRevision

Syntax DisplayString (SIZE(0..79))

Access read-only

Status mandatory

Description The port revision (e.g., for a GBIC).

Sequence ::= { connUnitPortEntry 13 }

connUnitPortVendor

Syntax DisplayString (SIZE(0..79))

Access read-only

Status mandatory

Description The port vendor (e.g., for a GBIC).

Sequence ::= { connUnitPortEntry 14 }

connUnitPortSpeed

Syntax INTEGER

Access read-only

Status mandatory

Description The speed of the port in kilobytes per second.

Sequence ::= { connUnitPortEntry 15 }

connUnitPortControl

Syntax INTEGER {
 unknown(1),
 invalid(2),
 resetConnUnitPort(3),

```
        bypassConnUnitPort(4),
        unbypassConnUnitPort(5),
        offlineConnUnitPort(6),
        onlineConnUnitPort(7),
        resetConnUnitPortCounters(8)
    }
Access      read-write
Status      mandatory
```

Description	<p>This object is used to control the addressed connUnit's port. Valid commands are:</p>
resetConnUnitPort:	<p>If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'reset' operation. Examples of these operations are: the Link Reset protocol, the Loop Initialization protocol, or a resynchronization occurring between the transceiver in the addressed port to the transceiver that the port is connected to.</p>
bypassConnUnitPort:	<p>If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'bypass' operation. Examples of these operations are transitioning from online to offline, a request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.</p>
unbypassConnUnitPort:	<p>If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'unbypass' operation. Examples of these operations are the Link Failure protocol, a request(PARTICIPATING) command to the Loop Port state machine, or addition of the port to an arbitrated loop by a hub.</p>

offlineConnUnitPort:	<p>If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'offline' operation. Examples of these operations are disabling a port's transceiver, the Link Failure protocol, request (NON-PARTICIPATING) command to the Loop Port state machine, or removal of the port from an arbitrated loop by a hub.</p>
onlineConnUnitPort:	<p>If the addressed connUnit allows this operation to be performed to this port, the addressed port performs a vendor-specific 'online' operation. Examples of these operations are enabling a port's transceiver, the Link Failure protocol, request (PARTICIPATING) command to the Loop Port state machine, or addition of the port from an arbitrated loop by a hub.</p>
resetConnUnitPortCounters:	<p>If the addressed connUnit allows this operation to be performed to this port, the addressed port statistics table counters will be set to zero.</p>

NOTE: Each implementation may chose not to allow any or all of these values on a SET. On a read, if you do not support write, then return invalid. Otherwise return the last control operation attempted.

Sequence ::= { connUnitPortEntry 16 }

connUnitPortName

Syntax	DisplayString
Access	read-write
Status	mandatory
Description	<p>A user-defined name for this port. This means that up to DisplayString characters may be supported. If less than, then the name will be truncated in the connunit.</p>

Sequence ::= { connUnitPortEntry 17 }

connUnitPortPhysicalNumber

Syntax INTEGER

Access read-only

Status mandatory

Description This is the internal port number this port is known by. In many implementations, this should be the same as connUnitPortIndex. Some implementations may have an internal port representation not compatible with the rules for table indices. In that case, provide the internal representation of this port in this object. This value may also be used in the connUnitLinkPortNumberX or connUnitLinkPortNumberY objects of the connUnitLinkTable.

Sequence ::= { connUnitPortEntry 18 }

connUnitPortStatObject

Syntax OBJECT IDENTIFIER

Access read-only

Status deprecated

Description This contains the OID of the first object of the table that contains the statistics for this particular port. If this has a value of zero, then there are no statistics available for this port. The port type information will help identify the statistics objects that will be found in the table.

Sequence ::= { connUnitPortEntry 19 }

connUnitPortProtocolCap

Syntax OCTET STRING (SIZE (2))

Access read-only

Status mandatory

Description Bit mask that specifies the driver level protocol capability of this port. If this is not applicable, return all bits set to zero.

The bits have the following definition:

unknown - 0

Loop - 1
 Fabric - 2
 SCSI - 4
 TCP/IP - 8
 VI - 16
 FICON - 32

Sequence ::= { connUnitPortEntry 20 }

connUnitPortProtocolOp

Syntax OCTET STRING (SIZE (2))
 Access read-only
 Status mandatory
 Description Bit mask that specifies the driver level protocol(s) that are currently operational. If this is not applicable, return all bits set to zero. This object has the same definition as connUnitPortProtocolCap.
 Sequence ::= { connUnitPortEntry 21 }

connUnitPortNodeWwn

Syntax FcNameId
 Access read-only
 Status mandatory
 Description The Node World Wide Name of the port if applicable, otherwise all zeros. This should have the same value for a group of related ports. The container is defined as the largest physical entity. For example, all ports on HBAs on a host will have the same Node WWN. All ports on the same storage subsystem will have the same Node WWN.
 Sequence ::= { connUnitPortEntry 22 }

connUnitPortHWState

Syntax	INTEGER { unknown (1), failed (2), bypassed (3), active (4), loopback (5), txfault (6), noMedia (7), linkDown (8)	port failed diagnostics FCAL bypass, loop only connected to a device Port in ext loopback Transmitter fault media not installed waiting for activity (rx sync) }
Access	read-only	
Status	mandatory	
Description	The hardware detected state of the port.	
Sequence	::= { connUnitPortEntry 23 }	

Event Group

connUnitEventTable

Syntax	SEQUENCE OF ConnUnitEventEntry
Access	not-accessible
Status	mandatory
Description	The table of connectivity unit events. Errors, warnings, and information should be reported in this table.
Sequence	::= { connSet 11 }

connUnitEventEntry

Syntax	ConnUnitEventEntry
Access	not-accessible
Status	mandatory

Description Each entry contains information on a specific event for the given connectivity unit.

INDEX { connUnitEventUnitId, connUnitEventIndex }

Sequence ::= { connUnitEventTable 1 }

ConnUnitEventEntry ::=

SEQUENCE {

connUnitEventUnitId

FcGlobalId,

connUnitEventIndex

INTEGER (1..2147483647),

connUnitEventId

INTEGER,

connUnitREventTime

DisplayString,

connUnitSEventTime

TimeTicks,

connUnitEventSeverity

FcEventSeverity,

connUnitEventType

INTEGER,

connUnitEventObject

OBJECT IDENTIFIER,

connUnitEventDescr

DisplayString }

connUnitEventUnitId

Syntax FcGlobalId

Access read-only

Status mandatory

Description The connUnitId of the connectivity unit that contains this event table.

Sequence ::= { connUnitEventEntry 1 }

connUnitEventIndex

Syntax INTEGER (1..2147483647)

Access read-only

Status mandatory

Description Each connectivity unit has its own event buffer. As it wraps, it may write over previous events. This object is an index into the buffer. It is recommended that this table be read using 'getNext's to retrieve the initial table. The management application should read the event table at periodic intervals and then determine if any new entries were added by comparing the last known index value with the current highest index value. The management application should then update its copy of the event table. If the read interval is too long, it is possible that there may be events that may not be contained in the agent's internal event buffer.

For example, an agent may read events 50-75. At the next read interval, connUnitEventCurrID is 189. If the management app tries to read event index 76, and the agent's internal buffer is 100 entries max, event index 76 will no longer be available.

The index value is an incrementing integer starting from one every time there is a table reset. On table reset, all contents are emptied and all indices are set to zero. When an event is added to the table, the event is assigned the next higher integer value than the last item entered into the table. If the index value reaches its maximum value, the next item entered will cause the index value to roll over and start at one again.

Sequence ::= { connUnitEventEntry 2 }

connUnitEventId

Syntax INTEGER

Access read-only

Status deprecated

Description The internal event Id. Incremented for each event, ranging between 1 and connUnitMaxEvents. Not used as table index to simplify the agent implementation. When this reaches the end of the range

specified by connUnitMaxEvents, the Id will roll over to start at one. This value will be set back to one at reset. The relationship of this value to the index is that internal event id may represent a smaller number than a 32 bit integer (e.g. max 100 entries) and would only have a value range up to connUnitMaxEvents.

Sequence ::= { connUnitEventEntry 3 }

connUnitREventTime

Syntax DisplayString (SIZE (0..15))

Access read-only

Status mandatory

Description This is the real time when the event occurred. It has the following format.

DDMMYYYY HHMMSS

DD=day number

MM=month number

YYYY=year number

HH=hour number

MM=minute number

SS=seconds number

If not applicable, return either a NULL string or '00000000 000000'.

Sequence ::= { connUnitEventEntry 4 }

connUnitSEventTime

Syntax TimeTicks

Access read-only

Status mandatory

Description This is the sysuptime timestamp when the event occurred.

Sequence ::= { connUnitEventEntry 5 }

connUnitEventSeverity

Syntax	FcEventSeverity
Access	read-only
Status	mandatory
Description	The event severity level.
Sequence	::= { connUnitEventEntry 6 }

connUnitEventType

Syntax	INTEGER { unknown(1), other(2), status(3), configuration(4), topology(5) }
Access	read-only
Status	mandatory
Description	The type of this event.
Sequence	::= { connUnitEventEntry 7 }

connUnitEventObject

Syntax	OBJECT IDENTIFIER
Access	read-only
Status	mandatory
Description	This is used with the connUnitEventType to identify which object the event refers to. Examples are connUnitPortStatus.connUnitId.connUnitPortIndex, connUnitStatus.connUnitId, etc.
Sequence	::= { connUnitEventEntry 8 }

connUnitEventDescr

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	The description of the event.
Sequence	::= { connUnitEventEntry 9 }

Link Table

This is intended to organize and communicate any information the agent possesses which would assist a management application to discover the CONNECTIVITY UNITS in the framework and the TOPOLOGY of their interconnect. That is, the goal is to assist the management application not only to LIST the elements of the framework, but to MAP them.

With this goal, the agent SHOULD include as much as it possesses about any links from its own connectivity units to others, including links among its own units.

An agent SHOULD include partial information about links if it is not able to fully define them. For an entry to be considered to be valid, both the X (local) and the Y (remote) need to have one valid value.

If the agent is able to discover links which do not directly attach to members of its agency and its discovery algorithm gives some assurance the links are recently valid, it MAY include these links.

Link information entered by administrative action MAY be included even if not validated directly if the link has at least one endpoint in this agency, but SHOULD NOT be included otherwise.

A connectivity unit should fill the table in as best it can. One of the methods to fill this in would be to use the RNID ELS (ANSI document 99-422v0). This allows one to query a port for the information needed for the link table.

This table is accessed either directly if the management software has an index value or via GetNexts. The value of the indexes are not required to be contiguous. Each entry created in this table will be assigned an index. This relationship is kept persistent until the entry is removed from the table or the system is reset. The total number of entries are defined by the size of the table.

connUnitLinkTable

Syntax	SEQUENCE OF ConnUnitLinkEntry
Access	not-accessible
Status	mandatory
Description	A list of links know to this agent from this connectivity unit to other connectivity units.
Sequence	::= { connSet 12 }

connUnitLinkEntry

Syntax	ConnUnitLinkEntry
Access	not-accessible
Status	mandatory
Description	An entry describing a particular link to another. INDEX { connUnitLinkUnitId, connUnitLinkIndex }
Sequence	::= { connUnitLinkTable 1 } ConnUnitLinkEntry ::= SEQUENCE { connUnitLinkUnitId FcGlobalId, connUnitLinkIndex INTEGER, connUnitLinkNodeIdX OCTET STRING, connUnitLinkPortNumberX INTEGER, connUnitLinkPortWwnX FcGlobalId, connUnitLinkNodeIdY OCTET STRING, connUnitLinkPortNumberY INTEGER, connUnitLinkPortWwnY FcGlobalId, connUnitLinkAgentAddressY OCTET STRING, connUnitLinkAgentAddressTypeY INTEGER, connUnitLinkAgentPortY INTEGER,

connUnitLinkUnitTypeY	FcUnitType,
connUnitLinkConnIdY	OCTET STRING,
connUnitLinkCurrIndex	INTEGER }

connUnitLinkUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this link table.
Sequence	::= { connUnitLinkEntry 1 }

connUnitLinkIndex

Syntax	INTEGER (1..2147483647)
Access	read-only
Status	mandatory
Description	This index is used to create a unique value for each entry in the link table with the same connUnitLinkUnitId. The value can only be reused if it is not currently in use and the value is the next candidate to be used. This value wraps at the highest value represented by the size of INTEGER. This value is reset to zero when the system is reset and the first value to be used is one.
Sequence	::= { connUnitLinkEntry 2 }

connUnitLinkNodeIdX

Syntax	OCTET STRING (SIZE(16))
Access	read-only
Status	mandatory
Description	The node WWN of the unit at one end of the link. If the node WWN is unknown and the node is a connUnit in the responding agent then the value of this object MUST BE equal to its connUnitID.
Sequence	::= { connUnitLinkEntry 3 }

connUnitLinkPortNumberX

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The port number on the unit specified by connUnitLinkNodeIdx if known, otherwise -1. If the value is nonnegative then it will be equal to connUnitPortPhysicalNumber.
Sequence	::= { connUnitLinkEntry 4 }

connUnitLinkPortWwnX

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The port WWN of the unit specified by connUnitLinkNodeIdx if known, otherwise 16 octets of binary 0.
Sequence	::= { connUnitLinkEntry 5 }

connUnitLinkNodeIdx

Syntax	OCTET STRING (SIZE(16))
Access	read-only
Status	mandatory
Description	The node WWN of the unit at the other end of the link. If the node WWN is unknown and the node is a connUnit in the responding SNMP agency then the value of this object MUST BE equal to its connUnitID.
Sequence	::= { connUnitLinkEntry 6 }

connUnitLinkPortNumberY

Syntax	INTEGER
Access	read-only
Status	mandatory

Description The port number on the unit specified by `connUnitLinkNodeIdY` if known, otherwise -1. If the value is nonnegative then it will be equal to `connUnitPortPhysicalNumber`.

Sequence ::= { `connUnitLinkEntry 7` }

connUnitLinkPortWwnY

Syntax `FcGlobalId`

Access read-only

Status mandatory

Description The port WWN on the unit specified by `connUnitLinkNodeIdY` if known, otherwise 16 octets of binary 0.

Sequence ::= { `connUnitLinkEntry 8` }

connUnitLinkAgentAddressY

Syntax `OCTET STRING (SIZE(16))`

Access read-only

Status mandatory

Description The address of an FCMGMT MIB agent for the node identified by `connUnitLinkNodeIdY`, if known; otherwise 16 octets of binary 0.

Sequence ::= { `connUnitLinkEntry 9` }

connUnitLinkAgentAddressTypeY

Syntax `INTEGER`

Access read-only

Status mandatory

Description If `connUnitLinkAgentAddressY` is nonzero, it is a protocol address. `ConnUnitLinkAgentAddressTypeY` is the 'address family number' assigned by IANA to identify the address format. (e.g., 1 is Ipv4, 2 is Ipv6). If `connUnitLinkAgentAddressY` is all zeros, then this value is ignored.

Sequence ::= { `connUnitLinkEntry 10` }

connUnitLinkAgentPortY

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The IP port number for the agent. This is provided in case the agent is at a non-standard SNMP port.
Sequence	::= { connUnitLinkEntry 11 }

connUnitLinkUnitTypeY

Syntax	FcUnitType
Access	read-only
Status	mandatory
Description	Type of the FC connectivity unit as defined in connUnitType.
Sequence	::= { connUnitLinkEntry 12 }

connUnitLinkConnIdY

Syntax	OCTET STRING (SIZE(3))
Access	read-only
Status	mandatory
Description	This is the Fibre Channel ID of this port. If the connectivity unit is a switch, this is expected to be a Big Endian value of 24 bits. If this is loop, then it is the ALPA that is connected. If this is an eport, then it will only contain the domain ID. If not any of those, unknown or cascaded loop, return all bits set to 1.
Sequence	::= { connUnitLinkEntry 13 }

connUnitLinkCurrIndex

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The last used link index.

Sequence ::= { connUnitLinkEntry 14 }

The following four tables have been obsoleted. These were used to keep statistic information based on the type of port type. It was changed for all ports to use a common statistics table.

connUnitPortStatHubTable

Syntax	SEQUENCE OF ConnUnitPortStatHubEntry
Access	not-accessible
Status	obsolete
Description	A list of statistics for the hub port type.
Sequence	::= { statSet 1 }

connUnitPortStatFabricTable

Syntax	SEQUENCE OF ConnUnitPortStatFabricEntry
Access	not-accessible
Status	obsolete
Description	A list of statistics for the fabric port types.
Sequence	::= { statSet 2 }

connUnitPortStatSCSITable

Syntax	SEQUENCE OF ConnUnitPortStatSCSIEntry
Access	not-accessible
Status	obsolete
Description	A list of statistics for the SCSI port type.
Sequence	::= { statSet 3 }

connUnitPortStatLANTable

Syntax	SEQUENCE OF ConnUnitPortStatLANEntry
Access	not-accessible
Status	obsolete

Description	A list of statistics for the LAN/WAN port type.
Sequence	::= { statSet 4 } There is one and only one statistics table for each individual port. For all objects in statistics table, if the object is not supported by the conn unit then the high order bit is set to 1 with all other bits set to zero. The high order bit is reserved to indicate if the object is supported or not. All objects start at a value of zero at hardware initialization and continue incrementing till end of 63 bits and then wrap to zero.

Port Statistics

connUnitPortStatTable

Syntax	SEQUENCE OF ConnUnitPortStatEntry
Access	not-accessible
Status	mandatory
Description	A list of statistics for the fabric port types.
Sequence	::= { statSet 5 }

connUnitPortStatEntry

Syntax	ConnUnitPortStatEntry
Access	not-accessible
Status	mandatory
Description	An entry describing port statistics. INDEX { connUnitPortStatUnitId, connUnitPortStatIndex }
Sequence	::= { connUnitPortStatTable 1 } ConnUnitPortStatEntry ::= SEQUENCE { connUnitPortStatUnitId FcGlobalId, connUnitPortStatIndex INTEGER,

connUnitPortStatCountError
OCTET STRING,
connUnitPortStatCountTxObjects
OCTET STRING,
connUnitPortStatCountRxObjects
OCTET STRING,
connUnitPortStatCountTxElements
OCTET STRING,
connUnitPortStatCountRxElements
OCTET STRING,
connUnitPortStatCountBBCreditZero
OCTET STRING,
connUnitPortStatCountInputBuffersFull
OCTET STRING,
connUnitPortStatCountFBSYFrames
OCTET STRING,
connUnitPortStatCountPBSYFrames
OCTET STRING,
connUnitPortStatCountFRJTFrames
OCTET STRING,
connUnitPortStatCountPRJTFrames
OCTET STRING,
connUnitPortStatCountClass1RxFrames
OCTET STRING,
connUnitPortStatCountClass1TxFrames
OCTET STRING,
connUnitPortStatCountClass1FBSYFrames
OCTET STRING,
connUnitPortStatCountClass1PBSYFrames

OCTET STRING,
connUnitPortStatCountClass1FRJTFrames
OCTET STRING,
connUnitPortStatCountClass1PRJTFrames
OCTET STRING,
connUnitPortStatCountClass2RxFrames
OCTET STRING,
connUnitPortStatCountClass2TxFrames
OCTET STRING,
connUnitPortStatCountClass2FBSYFrames
OCTET STRING,
connUnitPortStatCountClass2PBSYFrames
OCTET STRING,
connUnitPortStatCountClass2FRJTFrames
OCTET STRING,
connUnitPortStatCountClass2PRJTFrames
OCTET STRING,
connUnitPortStatCountClass3RxFrames
OCTET STRING,
connUnitPortStatCountClass3TxFrames
OCTET STRING,
connUnitPortStatCountClass3Discards
OCTET STRING,
connUnitPortStatCountRxMulticastObjects
OCTET STRING,
connUnitPortStatCountTxMulticastObjects
OCTET STRING,
connUnitPortStatCountRxBroadcastObjects
OCTET STRING,

connUnitPortStatCountTxBroadcastObjects
OCTET STRING,
connUnitPortStatCountRxLinkResets
OCTET STRING,
connUnitPortStatCountTxLinkResets
OCTET STRING,
connUnitPortStatCountNumberLinkResets
OCTET STRING,
connUnitPortStatCountRxOfflineSequences
OCTET STRING,
connUnitPortStatCountTxOfflineSequences
OCTET STRING,
connUnitPortStatCountNumberOfflineSequences
OCTET STRING,
connUnitPortStatCountLinkFailures
OCTET STRING,
connUnitPortStatCountInvalidCRC
OCTET STRING,
connUnitPortStatCountInvalidTxWords
OCTET STRING,
connUnitPortStatCountPrimitiveSequenceProtocolErrors
OCTET STRING,
connUnitPortStatCountLossofSignal
OCTET STRING,
connUnitPortStatCountLossofSynchronization
OCTET STRING,
connUnitPortStatCountInvalidOrderedSets
OCTET STRING,
connUnitPortStatCountFramesTooLong


```

        OCTET STRING,
connUnitPortStatCountFramesTruncated
        OCTET STRING,
connUnitPortStatCountAddressErrors
        OCTET STRING,
connUnitPortStatCountDelimiterErrors
        OCTET STRING,
connUnitPortStatCountEncodingDisparityErrors
        OCTET STRING }

```

connUnitPortStatUnitId

Syntax	FcGlobalId
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this port stat table.
Sequence	::= { connUnitPortStatEntry 1 }

connUnitPortStatIndex

Syntax	INTEGER (0..2147483647)
Access	read-only
Status	mandatory
Description	A unique value among all entries in this table, between 0 and connUnitNumPort[connUnitPortUnitId].
Sequence	::= { connUnitPortStatEntry 2 }

connUnitPortStatCountError

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory

Description	A count of the errors that have occurred on this port.
Sequence	::= { connUnitPortStatEntry 3 }

connUnitPortStatCountTxObjects

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	The number of frames/packets/IOs/etc. that have been transmitted by this port. Note: A Fibre Channel frame starts with SOF and ends with EOF. FC loop devices should not count frames passed through. This value represents the sum total for all other Tx objects.
Sequence	::= { connUnitPortStatEntry 4 }

connUnitPortStatCountRxObjects

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	The number of frames/packets/IOs/etc. that have been received by this port. Note: A Fibre Channel frame starts with SOF and ends with EOF. FC loop devices should not count frames passed through. This value represents the sum total for all other Rx objects.
Sequence	::= { connUnitPortStatEntry 5 }

connUnitPortStatCountTxElements

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	The number of octets or bytes that have been transmitted by this port. One second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput. Note, for Fibre Channel, ordered sets are not included in the count.
Sequence	::= { connUnitPortStatEntry 6 }

connUnitPortStatCountRxElements

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	The number of octets or bytes that have been received. by this port. One second periodic polling of the port. This value is saved and compared with the next polled value to compute net throughput. Note, for Fibre Channel, ordered sets are not included in the count.
Sequence	::= { connUnitPortStatEntry 7 }

connUnitPortStatCountBBCreditZero

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of transitions in/out of BBcredit zero state. The other side is not providing any credit. Note, this is a Fibre Channel stat only.
Sequence	::= { connUnitPortStatEntry 8 }

connUnitPortStatCountInputBuffersFull

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of occurrences when all input buffers of a port were full and outbound buffer-to-buffer credit transitioned to zero. There is no credit to provide to other side. Note, this is a Fibre Channel stat only.
Sequence	::= { connUnitPortStatEntry 9 }

connUnitPortStatCountFBSYFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory

Description Count of times that FBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. Port can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat. This is the sum of all classes. If you cannot keep the by class counters, then keep the sum counters.

Sequence ::= { connUnitPortStatEntry 10 }

connUnitPortStatCountPBSYFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that PBSY was returned to this port as a result of a frame that could not be delivered to the other end of the link. This occurs if the destination port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat. This is the sum of all classes. If you cannot keep the by class counters, then keep the sum counters.

Sequence ::= { connUnitPortStatEntry 11 }

connUnitPortStatCountFRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Frame that was rejected by the fabric. Note, This is the total for all classes and is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 12 }

connUnitPortStatCountPRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Frame that was rejected at the destination N_Port. Note, This is the total for all classes and is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 13 }

connUnitPortStatCountClass1RxFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Class 1 Frames received at this port. Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 14 }

connUnitPortStatCountClass1TxFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Class 1 Frames transmitted out this port. Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 15 }

connUnitPortStatCountClass1FBSYFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FBSY was returned to this port as a result of a Class 1 Frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 16 }

connUnitPortStatCountClass1PBSYFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that PBSY was returned to this port as a result of a Class 1 Frame that could not be delivered to the other end of the link. This occurs if the destination N_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 17 }

connUnitPortStatCountClass1FRJTFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that FRJT was returned to this port as a result of a Class 1 Frame that was rejected by the fabric. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 18 }

connUnitPortStatCountClass1PRJTFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that FRJT was returned to this port as a result of a Class 1 Frame that was rejected at the destination N_Port. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 19 }

connUnitPortStatCountClass2RxFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only

Status	mandatory
Description	Count of Class 2 Frames received at this port. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 20 }

connUnitPortStatCountClass2TxFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Class 2 Frames transmitted out this port. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 21 }

connUnitPortStatCountClass2FBSYFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that FBSY was returned to this port as a result of a Class 2 Frame that could not be delivered to the other end of the link. This occurs if either the Fabric or the destination port is temporarily busy. FBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 22 }

connUnitPortStatCountClass2PBSYFrames

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of times that PBSY was returned to this port as a result of a Class 2 Frame that could not be delivered to the other end of the link. This occurs if the destination N_Port is temporarily busy. PBSY can only occur on SOFc1 frames (the frames that establish a connection). Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 23 }

connUnitPortStatCountClass2FRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Class 2 Frame that was rejected by the fabric. Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 24 }

connUnitPortStatCountClass2PRJTFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of times that FRJT was returned to this port as a result of a Class 2 Frame that was rejected at the destination N_Port. Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 25 }

connUnitPortStatCountClass3RxFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of Class 3 Frames received at this port. Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 26 }

connUnitPortStatCountClass3TxFrames

Syntax OCTET STRING (SIZE (8))

Access read-only

Status	mandatory
Description	Count of Class 3 Frames transmitted out this port. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 27 }

connUnitPortStatCountClass3Discards

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Class 3 Frames that were discarded upon reception at this port. There is no FBSY or FRJT generated for Class 3 Frames. They are simply discarded if they cannot be delivered. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 28 }

connUnitPortStatCountRxMulticastObjects

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Multicast Frames or Packets received at this port.
Sequence	::= { connUnitPortStatEntry 29 }

connUnitPortStatCountTxMulticastObjects

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Multicast Frames or Packets transmitted out this port.
Sequence	::= { connUnitPortStatEntry 30 }

connUnitPortStatCountRxBroadcastObjects

Syntax	OCTET STRING (SIZE (8))
--------	-------------------------

Access	read-only
Status	mandatory
Description	Count of Broadcast Frames or Packets received at this port.
Sequence	::= { connUnitPortStatEntry 31 }

connUnitPortStatCountTxBroadcastObjects

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Broadcast Frames or Packets transmitted out this port. On a Fibre Channel loop, count only OPNr frames generated.
Sequence	::= { connUnitPortStatEntry 32 }

connUnitPortStatCountRxLinkResets

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Link resets. This is the number of LR's received. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 33 }

connUnitPortStatCountTxLinkResets

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Link resets. This is the number LR's transmitted. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 34 }

connUnitPortStatCountNumberLinkResets

Syntax	OCTET STRING (SIZE (8))
--------	-------------------------

Access	read-only
Status	mandatory
Description	Count of Link resets and LIPs detected at this port. The number times the reset link protocol is initiated. These are the count of the logical resets, a count of the number of primitives. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 35 }

connUnitPortStatCountRxOfflineSequences

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Offline Primitive OLS received at this port. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 36 }

connUnitPortStatCountTxOfflineSequences

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Offline Primitive OLS transmitted by this port. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 37 }

connUnitPortStatCountNumberOfflineSequences

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of Offline Primitive sequence received at this port. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 38 }

connUnitPortStatCountLinkFailures

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of link failures. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 39 }

connUnitPortStatCountInvalidCRC

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of frames received with invalid CRC. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Loop ports should not count CRC errors passing through when monitoring. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 40 }

connUnitPortStatCountInvalidTxWords

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of invalid transmission words received at this port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 41 }

connUnitPortStatCountPrimitiveSequenceProtocolErrors

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory

Description Count of primitive sequence protocol errors detected at this port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 42 }

connUnitPortStatCountLossofSignal

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of instances of signal loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 43 }

connUnitPortStatCountLossofSynchronization

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of instances of synchronization loss detected at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 44 }

connUnitPortStatCountInvalidOrderedSets

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description Count of invalid ordered sets received at port. This count is part of the Link Error Status Block (LESB). (FC-PH 29.8). Note, this is a Fibre Channel only stat.

Sequence ::= { connUnitPortStatEntry 45 }

connUnitPortStatCountFramesTooLong

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of frames received at this port where the frame length was greater than what was agreed to in FLOGI/PLOGI. This could be caused by losing the end of frame delimiter. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 46 }

connUnitPortStatCountFramesTruncated

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of frames received at this port where the frame length was less than the minimum indicated by the frame header - normally 24 bytes, but it could be more if the DFCTL field indicates an optional header should have been present. Note, this is a FC only stat.
Sequence	::= { connUnitPortStatEntry 47 }

connUnitPortStatCountAddressErrors

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of frames received with unknown addressing. e.x. unknown SID or DID. the SID or DID is not known to the routing algorithm. Note. this is a FC only stat.
Sequence	::= { connUnitPortStatEntry 48 }

connUnitPortStatCountDelimiterErrors

Syntax	OCTET STRING (SIZE (8))
Access	read-only

Status	mandatory
Description	Count of invalid frame delimiters received at this port. An example is a frame with a class 2 start and a class 3 at the end. Note, this is a FC only stat.
Sequence	::= { connUnitPortStatEntry 49 }

connUnitPortStatCountEncodingDisparityErrors

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	Count of disparity errors received at this port. Note, this is a Fibre Channel only stat.
Sequence	::= { connUnitPortStatEntry 50 }

FC Simple Name Server Table

The Fibre Channel Simple Name Server table contains an entry for each device presently known to this connUnit. There will not be any version on this since FC-GS3 does not define a version today.

This table is accessed either directly if the management software has an index value or via GetNexts. The value of the indexes are not required to be contiguous. Each entry created in this table will be assigned an index. This relationship is kept persistent until the entry is removed from the table or the system is reset. The total number of entries are defined by the size of the table

connUnitSnsMaxEntry

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The maximum number of entries in the table.
Sequence	::= { connUnitServiceScalars 1 }

connUnitSnsTable

Syntax	SEQUENCE OF ConnUnitSnsEntry
Access	not-accessible
Status	mandatory
Description	This table contains an entry for each object registered with this port in the switch.
Sequence	::= { connUnitServiceTables 1 }

connUnitSnsEntry

Syntax	ConnUnitSnsEntry
Access	not-accessible
Status	mandatory
Description	The Simple Name Server table for the port represented by ConnUnitSnsPortIndex . INDEX { connUnitSnsId, connUnitSnsPortIndex }
Sequence	::= { connUnitSnsTable 1 } ConnUnitSnsEntry ::= SEQUENCE { connUnitSnsId OCTET STRING, connUnitSnsPortIndex INTEGER, connUnitSnsPortIdentifier FcAddressId, connUnitSnsPortName FcNameId, connUnitSnsNodeName FcNameId, connUnitSnsClassOfSvc


```

        OCTET STRING,
connUnitSnsNodeIPAddress
        OCTET STRING,
connUnitSnsProcAssoc
        OCTET STRING,
connUnitSnsFC4Type
        OCTET STRING,
connUnitSnsPortType
        OCTET STRING,
connUnitSnsPortIPAddress
        OCTET STRING,
connUnitSnsFabricPortName
        FcNameId,
connUnitSnsHardAddress
        FcAddressId,
connUnitSnsSymbolicPortName
        DisplayString,
connUnitSnsSymbolicNodeName
        DisplayString }

```

connUnitSnsId

Syntax	OCTET STRING (SIZE (16))
Access	read-only
Status	mandatory
Description	The connUnitId of the connectivity unit that contains this Name Server table.
Sequence	::= { connUnitSnsEntry 1 }

connUnitSnsPortIndex

Syntax	INTEGER
--------	---------

Access	read-only
Status	mandatory
Description	The physical port number of this SNS table entry. Each physical port has an SNS table with 1-n entries indexed by ConnUnitSnsPortIdentifier (port address).
Sequence	::= { connUnitSnsEntry 2 }

connUnitSnsPortIdentifier

Syntax	FcAddressId
Access	read-only
Status	mandatory
Description	The Port Identifier for this entry in the SNS table.
Sequence	::= { connUnitSnsEntry 3 }

connUnitSnsPortName

Syntax	FcNameId
Access	read-only
Status	mandatory
Description	The Port WWN for this entry in the SNS table.
Sequence	::= { connUnitSnsEntry 4 }

connUnitSnsNodeName

Syntax	FcNameId
Access	read-only
Status	mandatory
Description	The Node Name for this entry in the SNS table.
Sequence	::= { connUnitSnsEntry 5 }

connUnitSnsClassOfSvc

Syntax	OCTET STRING (SIZE (1))
--------	-------------------------

Access	read-only
Status	mandatory
Description	The Classes of Service offered by this entry in the SNS table.
Sequence	::= { connUnitSnsEntry 6 }

connUnitSnsNodeIPAddress

Syntax	OCTET STRING (SIZE(16))
Access	read-only
Status	mandatory
Description	The IPv6 formatted address of the Node for this entry in the SNS table.
Sequence	::= { connUnitSnsEntry 7 }

connUnitSnsProcAssoc

Syntax	OCTET STRING (SIZE (1))
Access	read-only
Status	mandatory
Description	The Process Associator for this entry in the SNS table.
Sequence	::= { connUnitSnsEntry 8 }

connUnitSnsFC4Type

Syntax	OCTET STRING (SIZE (1))
Access	read-only
Status	mandatory
Description	The FC-4 Types supported by this entry in the SNS table.
Sequence	::= { connUnitSnsEntry 9 }

connUnitSnsPortType

Syntax	OCTET STRING (SIZE (1))
Access	read-only

Status	mandatory
Description	The Port Type of this entry in the SNS table.
Sequence	::= { connUnitSnsEntry 10 }

connUnitSnsPortIPAddress

Syntax	OCTET STRING (SIZE(16))
Access	read-only
Status	mandatory
Description	The IPv6 formatted address of this entry in the SNS table.
Sequence	::= { connUnitSnsEntry 11 }

connUnitSnsFabricPortName

Syntax	FcNameId
Access	read-only
Status	mandatory
Description	The Fabric Port name of this entry in the SNS table.
Sequence	::= { connUnitSnsEntry 12 }

connUnitSnsHardAddress

Syntax	FcAddressId
Access	read-only
Status	mandatory
Description	The Hard ALPA of this entry in the SNS table.
Sequence	::= { connUnitSnsEntry 13 }

connUnitSnsSymbolicPortName

Syntax	DisplayString (SIZE (0..79))
Access	read-only
Status	mandatory
Description	The Symbolic Port Name of this entry in the SNS table.

Sequence ::= { connUnitSnsEntry 14 }

connUnitSnsSymbolicNodeName

Syntax DisplayString (SIZE (0..79))
 Access read-only
 Status mandatory
 Description The Symbolic Node Name of this entry in the SNS table.
 Sequence ::= { connUnitSnsEntry 15 }

SNMP Trap Registration Group

trapMaxClients

Syntax INTEGER
 Access read-only
 Status mandatory
 Description The maximum number of SNMP trap recipients supported by the connectivity unit.
 Sequence ::= { trapReg 1 }

trapClientCount

Syntax INTEGER
 Access read-only
 Status mandatory
 Description The current number of rows in the trap table.
 Sequence ::= { trapReg 2 }

trapRegTable

Syntax SEQUENCE OF TrapRegEntry
 Access not-accessible
 Status mandatory

Description A table containing a row for each IP address/port number that traps will be sent to.

Sequence ::= { trapReg 3 }

trapRegEntry

Syntax TrapRegEntry

Access not-accessible

Status mandatory

Description IP/Port pair for a specific client.

INDEX { trapRegIpAddress,
trapRegPort }

Sequence ::= { trapRegTable 1 }
TrapRegEntry ::=
SEQUENCE {
trapRegIpAddress
IpAddress,
trapRegPort
INTEGER (1..2147483647),
trapRegFilter
FcEventSeverity,
trapRegRowState
INTEGER }

trapRegIpAddress

Syntax IpAddress

Access read-only

Status mandatory

Description The IP address of a client registered for traps.

Sequence ::= { trapRegEntry 1 }

trapRegPort

Syntax	INTEGER (1..2147483647)
Access	read-only
Status	mandatory
Description	The UDP port to send traps to for this host. Normally this would be the standard trap port (162). This object is an index and must be specified to create a row in this table.
Sequence	::= { trapRegEntry 2 }

trapRegFilter

Syntax	FcEventSeverity
Access	read-write
Status	mandatory
Description	This value defines the trap severity filter for this trap host. The connUnit will send traps to this host that have a severity level less than or equal to this value. The default value of this object is 'warning'.
Sequence	::= { trapRegEntry 3 }

trapRegRowState

Syntax	INTEGER { rowDestroy(1), Remove row from table. rowInactive(2), Row exists, but TRAPs disabled rowActive(3) Row exists and is enabled for sending traps }
Access	read-write
Status	mandatory
Description	Specifies the state of the row. rowDestroy READ: Can never happen. WRITE: Remove this row from the table.

rowInactive

READ: Indicates that this row does exist, but that traps are not enabled to be sent to the target.

WRITE: If the row does not exist, and the agent allows writes to the trap table, then a new row is created. The values of the optional columns will be set to default values. Traps are not enabled to be sent to the target. If the row already existed, then traps are disabled from being sent to the target.

rowActive

READ: Indicates that this row exists, and that traps are enabled to be sent to the target.

WRITE: If the row does not exist, and the agent allows writes to the trap table, then a new row is created. The values of the optional columns will be set to default values. Traps are enabled to be sent to the target. If the row already exists, then traps are enabled to be sent to the target.

A value of rowActive or rowInactive must be specified to create a row in the table.

Sequence ::= { trapRegEntry 4 }

Related Traps

connUnitStatusChange

Enterprise	fcmgmt
Variables	{ connUnitStatus, connUnitState }
Description	The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.
Type	FA MIB: Connection unit status change.
Summary	FA MIB: Connection unit status change.
Severity	Minor
Category	Status Events
TimeIndex	1
State	OPERATIONAL
Sequence	::= 1

NOTE: connUnitAddedTrap , 2, no longer used

connUnitDeletedTrap

Enterprise	fcmgmt
Variables	{ connUnitId }
Description	A connUnit has been deleted from this agent. Recommended severity level (for filtering): warning.
Type	FA MIB: Connection unit deleted.
Summary	FA MIB: Connection unit deleted.
Severity	Warning
Category	Status Events
TimeIndex	1
State	OPERATIONAL
Sequence	::= 3

connUnitEventTrap

Enterprise	fcmgmt
Variables	{ connUnitEventId, connUnitEventType, connUnitEventObject, connUnitEventDescr }
Description	An event has been generated by the connectivity unit. Recommended severity level (for filtering): info.
Type	FA MIB: Connection unit event trap.
Summary	FA MIB: Connection unit event trap.
Severity	Normal
Category	Configuration Events
TimeIndex	1
State	OPERATIONAL
Sequence	::= 4

connUnitSensorStatusChange

Enterprise	fcmgmt
Variables	{ connUnitSensorStatus }
Description	The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.
Type	FA MIB: Connection unit sensor status change.
Summary	FA MIB: Connection unit sensor status change.
Severity	Minor
Category	Status Events
TimeIndex	1
State	OPERATIONAL
Sequence	::= 5

connUnitPortStatusChange

Enterprise	fcmgmt
Variables	{ connUnitPortStatus, connUnitPortState }
Description	The overall status of the connectivity unit has changed. Recommended severity level (for filtering): alert.
Type	FA MIB: Port status change.
Summary	FA MIB: Port status change.
Severity	Minor
Category	Status Events
TimeIndex	99
State	DEGRADED
Sequence	::= 6 END

This appendix contains the McDATA Private Enterprise MIB (fceos.mib) listing. The fceos MIB, used by the Sphereon® Fabric switches and the Intrepid® 6064 and 6140 directors (updated to support zoning, port binding, threshold alerts, and open trunking). This listing is intended to be used as a reference, but to verify that it represents the latest version of the MIB, contact McDATA as described in [Where to Get Help](#) in the Preface of this manual.

FCEOS.MIB

CONTENTS = MIB Definition for Fibre Channel Switches running McDATA Enterprise Operating System™ firmware.

CHANGE HISTORY

Version	Description
---------	-------------

- | | |
|-----|---|
| 2.3 | Updated NPIV objects as not-accessible for 7.0, 7.1 and 7.2 |
| 2.2 | Added objects for XPM and NPIV features. |
| 2.1 | Added new FRU traps and link status change traps. |
| | Removed comment about fcEosCTACounter not implemented. |
| | Added 0 value for fcEosFruStatus. |
| | Changed fcEosFruPartNumber, fcEosFruSerialNumber, fcEosFruTestDate from DisplayString size 1..64 to 0..64 (IR #6012). |

Version	Description
2.0	Renamed from ef-6000.mib to fceos.mib and renamed all objects accordingly. Added new objects in support of FL_Ports (eosPortConfigType, eosPortOpType, eosPortALPAIndex, eosPortFAN, eosPortLIPsGenerated, eosPortLIPsDetected). Support of FL_Ports.
1.10	Added 64 bit counters and support for 2G
1.7	Added Port Binding, Threshold Alert, and Zoning objects.
1.3	Initial version.

FCEOS-MIB DEFINITIONS ::= BEGIN

IMPORTS

enterprises, Counter

FROM RFC1155-SMI

OBJECT-TYPE

FROM RFC-1212

TRAP-TYPE

FROM RFC-1215;

Textual conventions for this MIB

DisplayString ::= OCTET STRING

TruthValue ::= INTEGER { yes (1), no (2) }

FcEosSysOperStatus ::= INTEGER {

operational (1),

redundant-failure (2),

minor-failure (3),

major-failure (4),

not-operational (5) }

```

FcEosFruCode ::= INTEGER {
    fru-bkplane (1) , Backplane
    fru-ctp (2) ,      Control Processor card
    fru-sbar (3) ,      Serial Crossbar
    fru-fan2 (4) ,      Center fan module
    fru-fan (5) ,      Fan module
    fru-power (6) ,    Power supply module
    fru-reserved (7) , Reserved, not used
    fru-glsl (8) ,      Longwave, Single-Mode, LC connector, 1 Gig
    fru-gsml (9) ,      Shortwave, Multi-Mode, LC connector, 1 Gig
    fru-gxxl (10),      Mixed, LC connector, 1 Gig
    fru-gsf1 (11),      SFO pluggable, 1 Gig
    fru-gsf2 (12),      SFO pluggable, 2 Gig
    fru-glslr (13),      Longwave, Single-Mode, MT-RJ connector, 1 Gig
    fru-gsmr (14),      Shortwave, Multi-Mode, MT-RJ connector, 1 Gig
    fru-gxxr (15),      Mixed, MT-RJ connector, 1 Gig
    fru-fint1 (16),      F-Port, internal, 1 Gig
    fru-xpm (17)      XPM Port module, 10 Gig }

```

```

FcEosFruPosition ::= INTEGER (1..255)

```

```

FcEosPortIndex ::= INTEGER (1..2048)

```

```

FcEosPortPhyState ::= INTEGER {
    psNotInstalled (1),
    psAvailable (2),
    psBlocked (3),
    psUnavailable (4),
    psLinkFailure (5),
    psLinkFailLOL (6),
    psIntDiags (7),
    psExtLoop (8),
    psPortFail (9),

```

psSR (10),
psLR (11),
psInaccessible (12),
psInactive (13),
psUnaddressable (14),
psDegraded (15),
psDisabled (16),
psInvalidAttach (17),
psSegmented (18),
other (19) }

FcEosPortWWN	::= OCTET STRING (SIZE (8))
FcEosPortList	::= OCTET STRING (SIZE (32))
LoopPortALPA	::= OCTET STRING (SIZE (16))
VirtualPortNPIV	::= OCTET STRING (SIZE(32))

Enterprise Specific Object Identifiers

mcData	OBJECT IDENTIFIER	::= { enterprises 289 } Product lines or generic product information
common	OBJECT IDENTIFIER	::= { mcData 1 }
commDev	OBJECT IDENTIFIER	::= { mcData 2 } -- communication devices

Fibre Channel product lines

fibreChannel	OBJECT IDENTIFIER	::= { commDev 1 }
fcSwitch	OBJECT IDENTIFIER	::= { fibreChannel 1 }
fcEos	OBJECT IDENTIFIER	::= { fcSwitch 2 }

Groups in FCEOS MIB

f		
cEosSys	OBJECT IDENTIFIER	::= { fcEos 1 }
fcEosFru	OBJECT IDENTIFIER	::= { fcEos 2 }
fcEosPort	OBJECT IDENTIFIER	::= { fcEos 3 }
fcEosPortBinding	OBJECT IDENTIFIER	::= { fcEos 4 }
fcEosZoning	OBJECT IDENTIFIER	::= { fcEos 5 }
fcEosTA	OBJECT IDENTIFIER	::= { fcEos 6 }
reserved	OBJECT IDENTIFIER	::= { fcEos 6 }
reserved	OBJECT IDENTIFIER	::= { fcEos 7 }
reserved	OBJECT IDENTIFIER	::= { fcEos 8 }
reserved	OBJECT IDENTIFIER	::= { fcEos 9 }

System Group

fcEosSysCurrentDate

Syntax	DisplayString (SIZE (1..64))
Access	read-only
Status	mandatory
Description	The current date information.
Sequence	::= { fcEosSys 1 }

fcEosSysBootDate

Syntax	DisplayString (SIZE (1..64))
Access	read-only
Status	mandatory
Description	The date and time of the last IPL of the switch.
Sequence	::= { fcEosSys 2 }

fcEosSysFirmwareVersion

Syntax	DisplayString (SIZE (1..24))
Access	read-only
Status	mandatory
Description	The current version of the firmware.
Sequence	::= { fcEosSys 3 }

fcEosSysTypeNum

Syntax	DisplayString (SIZE (1..64))
Access	read-only
Status	mandatory
Description	The ASCII type number of the switch.
Sequence	::= { fcEosSys 4 }

fcEosSysModelNum

Syntax	DisplayString (SIZE (1..64))
Access	read-only
Status	mandatory
Description	The ASCII model number of the switch.
Sequence	::= { fcEosSys 5 }

fcEosSysMfg

Syntax	DisplayString (SIZE (1..64))
Access	read-only
Status	mandatory
Description	The ASCII manufacturer of the switch.
Sequence	::= { fcEosSys 6 }

fcEosSysPlantOfMfg

Syntax	DisplayString (SIZE (1..64))
Access	read-only
Status	mandatory
Description	The ASCII plant of manufacturer of the switch.
Sequence	::= { fcEosSys 7 }

fcEosSysEcLevel

Syntax	DisplayString (SIZE (1..64))
Access	read-only
Status	mandatory
Description	The ASCII EC level ID of the switch.
Sequence	::= { fcEosSys 8 }

fcEosSysSerialNum

Syntax	DisplayString (SIZE (1..64))
Access	read-only
Status	mandatory
Description	The ASCII system serial number of the switch.
Sequence	::= { fcEosSys 9 }

fcEosSysOperStatus

Syntax	FcEosSysOperStatus
Access	read-only
Status	mandatory
Description	The current operational status of the switch.
Sequence	::= { fcEosSys 10 }

fcEosSysState

Syntax	INTEGER { online(1), coming-online(2), offline(3), going-offline(4) }
Access	read-only
Status	mandatory
Description	If the operational status of the switch is operational, the switch will be in one of the four states: online(1), coming-online(2), offline(3), and going-offline(4).
Sequence	::= { fcEosSys 11 }

fcEosSysAdmStatus

Syntax	INTEGER { online (1), offline (2) }
Access	read-write
Status	mandatory
Description	The desired administrative status of the switch. A management station may place the switch in a desired state by setting this object accordingly. The desired administrative status are online(1) and offline(2). The online means setting the switch to be accessible by an

external Fibre Channel port, and offline means setting the switch to be inaccessible.

Sequence ::= { fcEosSys 12 }

fcEosSysConfigSpeed

Syntax INTEGER { one-gig (1), two-gig (2) }

Access read-write

Status mandatory

Description Switch speed capability. It's a user initiated option to adjust the system-wide port speed capability. This object is supported for 2G capable 6064 Director only.

Sequence ::= { fcEosSys 13 }

fcEosSysOpenTrunking

Syntax TruthValue

Access read-write

Status mandatory

Description This object identifies/configures if McDATA Open Trunking is enabled or not.

Sequence ::= { fcEosSys 14 }

fcEosSysSwitchName

Syntax DisplayString (SIZE (1..64))

Access read-write

Status mandatory

Description The ASCII name of the switch.

Sequence ::= { fcEosSys 15 }

fcEosSysSwitchId

Syntax OCTET STRING (SIZE (8))

Access read-only

Status	mandatory
Description	The Worldwide Name of the switch.
Sequence	::= { fcEosSys 16 }

fcEosSysNPIV

Syntax	TruthValue
Access	not-accessible
Status	mandatory
Description	This object identifies/configures if McDATA NPIV feature is enabled or not.
Sequence	::= { fcEosSys 17 }

Fibre Channel FRU Group

This group contains FRU information of each Fibre Channel Module.

fcEosFruTable

Syntax	SEQUENCE OF FcEosFruEntry
Access	not-accessible
Status	mandatory
Description	A table that contains one entry for each module.
Sequence	::= { fcEosFru 1 }

fcEosFruEntry

Syntax	FcEosFruEntry
Access	not-accessible
Status	mandatory
Description	An entry containing the service parameters of the module. INDEX { fcEosFruCode, fcEosFruPosition }
Sequence	::= { fcEosFruTable 1 }

```

FcEosFruEntry ::= SEQUENCE {
    fcEosFruCode          FcEosFruCode,
    fcEosFruPosition      FcEosFruPosition,
    fcEosFruStatus        INTEGER,
    fcEosFruPartNumber    DisplayString,
    fcEosFruSerialNumber  DisplayString,
    fcEosFruPowerOnHours  Counter,
    fcEosFruTestDate      DisplayString }

```

fcEosFruCode

Syntax	FcEosFruCode
Access	read-only
Status	mandatory
Description	Field Replaceable Unit. A hardware component of the product that is replaceable as an entire unit. Each module defined in this MIB has a fixed FRU code.
Sequence	::= { fcEosFruEntry 1 }

fcEosFruPosition

Syntax	FcEosFruPosition
Access	read-only
Status	mandatory
Description	This object identifies the position of the module. The value starts from 1 to the maximum number of the cards that can be contained within this switch.
Sequence	::= { fcEosFruEntry 2 }

fcEosFruStatus

Syntax	INTEGER { unknown(0), active(1),
--------	--

	backup(2), update-busy(3), failed(4) }
Access	read-only
Status	mandatory
Description	This object identifies the operational status of the module. The unknown(0) state indicates no information is known about the module, the active(1) state indicates that the current module is active; The backup(2) state indicates that the back up module is used; The update-busy (3) state indicates that the module is in the updating process; The failed(4) state indicates that the current module is failed.
Sequence	::= { fcEosFruEntry 3 }

fcEosFruPartNumber

Syntax	DisplayString (SIZE (0..64))
Access	read-only
Status	mandatory
Description	The part number of the module.
Sequence	::= { fcEosFruEntry 4 }

fcEosFruSerialNumber

Syntax	DisplayString (SIZE (0..64))
Access	read-only
Status	mandatory
Description	The serial number of the module.
Sequence	::= { fcEosFruEntry 5 }

fcEosFruPowerOnHours

Syntax	Counter
Access	read-only
Status	mandatory

Description	The number of the hours that the FRU has been in operation.
Sequence	::= { fcEosFruEntry 6 }

fcEosFruTestDate

Syntax	DisplayString (SIZE (0..64))
Access	read-only
Status	mandatory
Description	The final test date of the module.
Sequence	::= { fcEosFruEntry 7 }

Fibre Channel Port Group

This group contains information about each Fibre Channel port.

fcEosPortTable

Syntax	SEQUENCE OF FcEosPortEntry
Access	not-accessible
Status	mandatory
Description	A table that contains one entry for each switch port.
Sequence	::= { fcEosPort 1 }

fcEosPortEntry TYPE

Syntax	FcEosPortEntry
Access	not-accessible
Status	mandatory
Description	An entry containing the information of the switch port. INDEX { fcEosPortIndex }
Sequence	::= { fcEosPortTable 1 }

FcEosPortEntry ::= SEQUENCE {

fcEosPortIndex	FcEosPortIndex,
fcEosPortPhyState	FcEosPortPhyState,
fcEosPortOpStatus	INTEGER,
fcEosPortAdmStatus	INTEGER,
fcEosPortConnector	INTEGER,
fcEosPortDistance	INTEGER,
fcEosPortXceiverType	INTEGER,
fcEosPortMedia	INTEGER,
fcEosPortSpeedCap	INTEGER,
fcEosPortConfigSpeed	INTEGER,
fcEosPortOpSpeed	INTEGER,
fcEosPortConfigType	INTEGER,
fcEosPortOpType	INTEGER,
fcEosPortALPAIndex	LoopPortALPA,
fcEosPortFAN	TruthValue,

Throughput statistics (32 bit counters)

fcEosPortTxWords32	Counter,
fcEosPortRxWords32	Counter,
fcEosPortTxFrames32	Counter,
fcEosPortRxFrames32	Counter,
fcEosPortTxThroughput	Counter,
fcEosPortRxThroughput	Counter,

Throughput statistics (32 bit counters)

fcEosPortTxWords32	Counter,
fcEosPortRxWords32	Counter,
fcEosPortTxFrames32	Counter,
fcEosPortRxFrames32	Counter,
fcEosPortTxThroughput	Counter,
fcEosPortRxThroughput	Counter,

Class 2 statistics (32 bit counters)

fcEosPortTxC2Words32	Counter,
fcEosPortRxC2Words32	Counter,
fcEosPortTxC2Frames32	Counter,
fcEosPortRxC2Frames32	Counter,
fcEosPortTxC2Octets32	Counter,
fcEosPortRxC2Octets32	Counter,
fcEosPortRxC2FabricReject32	Counter,
fcEosPortRxC2FabricBusy32	Counter,

Class 3 statistics (32 bit counters)

fcEosPortTxC3Words32	Counter,
fcEosPortRxC3Words32	Counter,
fcEosPortTxC3Frames32	Counter,
fcEosPortRxC3Frames32	Counter,
fcEosPortTxC3Octets32	Counter,
fcEosPortRxC3Octets32	Counter,
fcEosPortC3Discards32	Counter,

Operation statistics (32 bit counters)

fcEosPortDiscardFrames	Counter,
fcEosPortTxLinkResets	Counter,
fcEosPortRxLinkResets	Counter,
fcEosPortTxOLs	Counter,
fcEosPortRxOLs	Counter,
fcEosPortLIPsGenerated	Counter,
fcEosPortLIPsDetected	Counter,

Error statistics (32 bit counters)

fcEosPortAddrIDErrors	Counter,
fcEosPortDelimiterErrors	Counter,
fcEosPortSyncLosses	Counter,
fcEosPortSigLosses	Counter,
fcEosPortProtocolErrors	Counter,
fcEosPortInvalidTxWords	Counter,
fcEosPortLinkFailures	Counter,
fcEosPortCrcs	Counter,
fcEosPortTruncs	Counter,

Throughput statistics (64 bit counters)

fcEosPortTxWords64	OCTET STRING,
fcEosPortRxWords64	OCTET STRING,
fcEosPortTxFrames64	OCTET STRING,
fcEosPortRxFrames64	OCTET STRING,

Class 2 statistics (64 bit counters)

fcEosPortTxC2Words64	OCTET STRING,
fcEosPortRxC2Words64	OCTET STRING,
fcEosPortTxC2Frames64	OCTET STRING,
fcEosPortRxC2Frames64	OCTET STRING,
fcEosPortTxC2Octets64	OCTET STRING,
fcEosPortRxC2Octets64	OCTET STRING,

Class 3 statistics (64 bit counters)

fcEosPortTxC3Words64	OCTET STRING,
fcEosPortRxC3Words64	OCTET STRING,
fcEosPortTxC3Frames64	OCTET STRING,
fcEosPortRxC3Frames64	OCTET STRING,

fcEosPortTxC3Octets64	OCTET STRING,
fcEosPortRxC3Octets64	OCTET STRING,
fcEosPortC3Discards64	OCTET STRING,

Trunking statistics (32 bit counters)

fcEosPortTxFlows	Counter,
fcEosPortRxFlows	Counter,

Link incident information

fcEosPortLinkTrapEnable	TruthValue,
fcEosPortLinkEvent	INTEGER,
fcEosPortLinkEventTime	DisplayString,
fcEosPortName	DisplayString,
fcEosPortWWN	FcEosPortWWN,

N Port Virtualization information

fcEosPortNPiVIndex	VirtualPortNPiV,
fcEosPortNPiVMaxLogins	INTEGER }

fcEosPortIndex

Syntax	FcEosPortIndex
Access	read-only
Status	mandatory
Description	The fixed physical port number on the switch. It ranges from 1 to the number of physical ports that can be supported in the switch.
Sequence	::= { fcEosPortEntry 1 }

fcEosPortPhyState

Syntax	FcEosPortPhyState
Access	read-only
Status	mandatory

Description	The physical state of the port.
Sequence	::= { fcEosPortEntry 2 }
<hr/>	
fcEosPortOpStatus	
Syntax	INTEGER { online (1), offline (2), testing (3), faulty (4) }
Access	read-only
Status	mandatory
Description	The operational status of the port. The online(1) state indicates that user frames can be passed.
Sequence	::= { fcEosPortEntry 3 }
<hr/>	
fcEosPortAdmStatus	
Syntax	INTEGER { online (1), offline (2), testing (3) }
Access	read-write
Status	mandatory
Description	The desired state of the port. A management station may place the port in a desired state by setting this object accordingly. The testing(3) state indicates that no user frames can be passed. As the result of either explicit management action or per configuration information accessible by the switch, fcEosPortAdmStatus is then changed to either the online(1) or testing(3) states, or remains in the offline state.
Sequence	::= { fcEosPortEntry 4 }
<hr/>	
fcEosPortConnector	
Syntax	INTEGER { unknown (1), lc (2), mt-rj (3), mu (4), internal-port (5) }
Access	read-only
Status	mandatory
Description	Supported connector types of the port.
Sequence	::= { fcEosPortEntry 5 }

fcEosPortDistance

Syntax	INTEGER (0..255)
Access	read-only
Status	mandatory
Description	A bit map to represent distance types of the Port. bit 0 unknown bit 1-3 reserved bit 4 long distance (l) bit 5 intermediate distance (i) bit 6 short distance (s) bit 7 very long distance.
Sequence	::= { fcEosPortEntry 6 }

fcEosPortXceiverType

Syntax	INTEGER { unknown (1), longDistance (2), (LL-V) longWaveLaser-LL (3), -- (LL) shortWaveLaser-OFC (4), -- (SL) shortWaveLaser-noOFC (5), -- (SN) longWaveLaser-LC (6) -- (LC) }
Access	read-only
Status	mandatory
Description	The type of the installed transceiver.
Sequence	::= { fcEosPortEntry 7 }

fcEosPortMedia

Syntax	INTEGER (0..255)
Access	read-only

Status	mandatory
Description	A bit map to represent the media of the installed transceiver.
bit 0	single mode (sm)
bit 1	reserved
bit 2	multi-mode, 50m (m5)
bit 3	multi-mode, 62.5 (m6)
bit 4-6	reserved
bit 7	copper
Sequence	::= { fcEosPortEntry 8 }

fcEosPortSpeedCap

Syntax	INTEGER (0..255)
Access	read-only
Status	mandatory
Description	A bit map to represent the speed capability of the installed transceiver.
bit 0	100 MBytes/Sec
bit 1	reserved
bit 2	200 MBytes/Sec
bit 3	reserved
bit 4	400 MBytes/Sec
bit 5	reserved
bit 6	1000 MBytes/Sec
bit 7	reserved
Sequence	::= { fcEosPortEntry 9 }

fcEosPortConfigSpeed

Syntax	INTEGER { one-gig (1), two-gig (2), negotiate (3),
--------	---

	four-gig (4), ten-gig (10) }
Access	read-write
Status	mandatory
Description	The configured port speed.
Sequence	::= { fcEosPortEntry 10 }

fcEosPortOpSpeed

Syntax	INTEGER { unknown (1), one-gig (2), two-gig (3), negotiate (4), four-gig (5), ten-gig (10) }
Access	read-only
Status	mandatory
Description	The operating port speed.
Sequence	::= { fcEosPortEntry 11 }

fcEosPortConfigType

Syntax	INTEGER { gPort (1), fPort (2), ePort (3), flPort (4), fxPort (5), gxPort (6) }
Access	read-write

Status	mandatory
Description	The configured port type.
Sequence	::= { fcEosPortEntry 12 }

fcEosPortOpType

Syntax	INTEGER { unknown (1), ePort (2), fPort (3), flPort (4) }
Access	read-only
Status	mandatory
Description	The operating port type.
Sequence	::= { fcEosPortEntry 13 }

fcEosPortALPAIndex

Syntax	LoopPortALPA
Access	read-only
Status	mandatory
Description	The ALPA-Index bit map that identifies the list of ALPA's associated with the FL_port. Only applicable for flPorts.
Sequence	::= { fcEosPortEntry 14 }

fcEosPortFAN

Syntax	TruthValue
Access	read-write
Status	mandatory
Description	This object identifies/configures if the port supports Fabric Address Notification mode. Only applicable for flPorts.
Sequence	::= { fcEosPortEntry 15 }

Throughput statistics (32 bit counters)

fcEosPortTxWords32

Syntax	Counter
--------	---------

Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of words within frames that the port has transmitted. (Primitive signals and primitive sequence are not included.)
Sequence	::= { fcEosPortEntry 20 }

fcEosPortRxWords32

Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of words within frames that the port has received. (Primitive signals and primitive sequence are not included.)
Sequence	::= { fcEosPortEntry 21 }

fcEosPortTxFrames32

Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of (Fibre Channel) frames that the port has transmitted.
Sequence	::= { fcEosPortEntry 22 }

fcEosPortRxFrames32

Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of (Fibre Channel) frames that the port has received.
Sequence	::= { fcEosPortEntry 23 }

fcEosPortTxThroughput

Syntax	Counter
Access	read-only
Status	mandatory
Description	The Bps (bytes per second) transmission rate of the port.
Sequence	::= { fcEosPortEntry 24 }

fcEosPortRxThroughput

Syntax	Counter
Access	read-only
Status	mandatory
Description	The Bps (bytes per second) reception rate of the port.
Sequence	::= { fcEosPortEntry 25 }

Class 2 statistics (32 bit counters)

fcEosPortTxC2Words32

Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of class 2 4-byte words that the port has transmitted. (Primitive signals and primitive sequence are not included.)
Sequence	::= { fcEosPortEntry 30 }

fcEosPortRxC2Words32

Syntax	Counter
Access	read-only
Status	mandatory

Description A 32 bit counter for the number of class 2 4-byte words that the port has received. (Primitive signals and primitive sequence are not included.)

Sequence ::= { fcEosPortEntry 31 }

fcEosPortTxC2Frames32

Syntax Counter

Access read-only

Status mandatory

Description A 32 bit counter for the number of Class 2 frames that the port has transmitted

Sequence ::= { fcEosPortEntry 32 }

fcEosPortRxC2Frames32

Syntax Counter

Access read-only

Status mandatory

Description A 32 bit counter for the number of Class 2 frames that the port has received.

Sequence ::= { fcEosPortEntry 33 }

fcEosPortTxC2Octets32

Syntax Counter

Access read-only

Status mandatory

Description A 32 bit counter for the number of Class 2 Octets that the port has transmitted.

Sequence ::= { fcEosPortEntry 34 }

fcEosPortRxC2Octets32

Syntax Counter

Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of Class 2 Octets that the port has received.
Sequence	::= { fcEosPortEntry 35 }

fcEosPortRxC2FabricReject32

Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of Class 2 fabric rejects.
Sequence	::= { fcEosPortEntry 36 }

fcEosPortRxC2FabricBusy32

Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of Class 2 fabric busies.
Sequence	::= { fcEosPortEntry 37 }

Class 3 statistics (32 bit counters)

fcEosPortTxC3Words32

Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of class 3 4-byte words that the port has transmitted. (Primitive signals and primitive sequence are not included.)
Sequence	::= { fcEosPortEntry 40 }

fcEosPortRxC3Words32

Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of class 3 4-byte words that the port has received. (Primitive signals and primitive sequence are not included.)
Sequence	::= { fcEosPortEntry 41 }

fcEosPortTxC3Frames32

Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of Class 3 frames that the port has transmitted.
Sequence	::= { fcEosPortEntry 42 }

fcEosPortRxC3Frames32

Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of Class 3 frames that the port has received.
Sequence	::= { fcEosPortEntry 43 }

fcEosPortTxC3Octets32

Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of Class 3 Octets that the port has transmitted.

Sequence	::= { fcEosPortEntry 44 }
<hr/>	
fcEosPortRxC3Octets32	
Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of Class 3 Octets that the port has received.
Sequence	::= { fcEosPortEntry 45 }

<hr/>	
fcEosPortC3Discards32	
Syntax	Counter
Access	read-only
Status	mandatory
Description	A 32 bit counter for the number of Class 3 frames that the port has discarded.
Sequence	::= { fcEosPortEntry 46 }

Operation statistics (32 bit counters)

<hr/>	
fcEosPortDiscardFrames	
Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of frames that the port has discarded.
Sequence	::= { fcEosPortEntry 50 }

<hr/>	
fcEosPortTxLinkResets	
Syntax	Counter
Access	read-only
Status	mandatory

Description	The number of link resets initiated by this switch port.
Sequence	::= { fcEosPortEntry 51 }

fcEosPortRxLinkResets

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of link resets initiated by the attached N_port.
Sequence	::= { fcEosPortEntry 52 }

fcEosPortTxOLSS

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of offline sequences initiated by this switch port.
Sequence	::= { fcEosPortEntry 53 }

fcEosPortRxOLSS

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of offline sequences initiated by the attached N_port.
Sequence	::= { fcEosPortEntry 54 }

fcEosPortLIPsGenerated

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of LIPs generated/initiated/sent by this port. Only applicable for flPort.

Sequence	::= { fcEosPortEntry 55 }
<hr/>	
fcEosPortLIPsDetected	
Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of LIPs detected/received by this port. Only applicable for flPort.
Sequence	::= { fcEosPortEntry 56 }

Error statistics (32 bit counters)

<hr/>	
fcEosPortAddrIDErrors	
Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of address ID errors.
Sequence	::= { fcEosPortEntry 58 }

<hr/>	
fcEosPortDelimiterErrors	
Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of delimiter errors.
Sequence	::= { fcEosPortEntry 59 }

<hr/>	
fcEosPortSyncLosses	
Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of loss of synchronization timeouts.

Sequence ::= { fcEosPortEntry 60 }

fcEosPortSigLosses

Syntax Counter
 Access read-only
 Status mandatory
 Description The number of times that a Loss of Signal is detected.
 Sequence ::= { fcEosPortEntry 61 }

fcEosPortProtocolErrors

Syntax Counter
 Access read-only
 Status mandatory
 Description The number of protocol errors detected.
 Sequence ::= { fcEosPortEntry 62 }

fcEosPortInvalidTxWords

Syntax Counter
 Access read-only
 Status mandatory
 Description The number of Invalid Transmission words that the port has received.
 Sequence ::= { fcEosPortEntry 63 }

fcEosPortLinkFailures

Syntax Counter
 Access read-only
 Status mandatory
 Description The number of transitions to an LFX state.
 Sequence ::= { fcEosPortEntry 64 }

fcEosPortCrcs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of CRC errors detected from frames received.
Sequence	::= { fcEosPortEntry 65 }

fcEosPortTruncs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of frames shorter than the Fibre Channel minimum.
Sequence	::= { fcEosPortEntry 66 }

Throughput statistics (64 bit counters)

fcEosPortTxWords64

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	A 64 bit counter for the number of words within frames that the port has transmitted. (Primitive signals and primitive sequence are not included.)
Sequence	::= { fcEosPortEntry 67 }

fcEosPortRxWords64

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory

Description A 64 bit counter for the number of words within frames that the port has received. (Primitive signals and primitive sequence are not included.)

Sequence ::= { fcEosPortEntry 68 }

fcEosPortTxFrames64

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description A 64 bit counter for the number of (Fibre Channel) frames that the port has transmitted.

Sequence ::= { fcEosPortEntry 69 }

fcEosPortRxFrames64

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description A 64 bit counter for the number of (Fibre Channel) frames that the port has received.

Sequence ::= { fcEosPortEntry 70 }

Class 2 statistics (64 bit counters)

fcEosPortTxC2Words64

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description A 64 bit counter for the number of class 2 4-byte words that the port has transmitted. (Primitive signals and primitive sequence are not included.)

Sequence ::= { fcEosPortEntry 71 }

fcEosPortRxC2Words64

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	A 64 bit counter for the number of class 2 4-byte words that the port has received. (Primitive signals and primitive sequence are not included.)
Sequence	::= { fcEosPortEntry 72 }

fcEosPortTxC2Frames64

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	A 64 bit counter for the number of Class 2 frames that the port has transmitted.
Sequence	::= { fcEosPortEntry 73 }

fcEosPortRxC2Frames64

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	A 64 bit counter for the number of Class 2 frames that the port has received.
Sequence	::= { fcEosPortEntry 74 }

fcEosPortTxC2Octets64

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	A 64 bit counter for the number of Class 2 Octets that the port has transmitted.

Sequence ::= { fcEosPortEntry 75 }

fcEosPortRxC2Octets64

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description A 64 bit counter for the number of Class 2 Octets that the port has received.

Sequence ::= { fcEosPortEntry 76 }

Class 3 statistics (64 bit counters)

fcEosPortTxC3Words64

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description A 64 bit counter for the number of class 3 4-byte words that the port has transmitted. (Primitive signals and primitive sequence are not included.)

Sequence ::= { fcEosPortEntry 77 }

fcEosPortRxC3Words64

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description A 64 bit counter for the number of class 3 4-byte words that the port has received. (Primitive signals and primitive sequence are not included.)

Sequence ::= { fcEosPortEntry 78 }

fcEosPortTxC3Frames64

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	A 64 bit counter for the number of Class 3 frames that the port has transmitted.
Sequence	::= { fcEosPortEntry 79 }

fcEosPortRxC3Frames64

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	A 64 bit counter for the number of Class 3 frames that the port has received.
Sequence	::= { fcEosPortEntry 80 }

fcEosPortTxC3Octets64

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	A 64 bit counter for the number of Class 3 Octets that the port has transmitted.
Sequence	::= { fcEosPortEntry 81 }

fcEosPortRxC3Octets64

Syntax	OCTET STRING (SIZE (8))
Access	read-only
Status	mandatory
Description	A 64 bit counter for the number of Class 3 Octets that the port has received.

Sequence ::= { fcEosPortEntry 82 }

fcEosPortC3Discards64

Syntax OCTET STRING (SIZE (8))

Access read-only

Status mandatory

Description A 64 bit counter for the number of Class 3 frames that the port has discarded.

Sequence ::= { fcEosPortEntry 83 }

Trunking statistics (32 bit counters)

fcEosPortTxFlows

Syntax Counter

Access read-only

Status mandatory

Description The number of flows rerouted from this port.

Sequence ::= { fcEosPortEntry 100 }

fcEosPortRxFlows

Syntax Counter

Access read-only

Status mandatory

Description The number of flows rerouted to this port.

Sequence ::= { fcEosPortEntry 101 }

fcEosPortLinkTrapEnable

Syntax TruthValue

Access read-write

Status mandatory

Description This object indicates whether link event traps are enabled or disabled for this port. The value of this object does not affect the port status change traps.

Sequence ::= { fcEosPortEntry 140 }

fcEosPortLinkEvent

Syntax INTEGER {
 bit-error (1),
 loss-of-signal (2),
 nos-received (3),
 link-failure (4),
 invalid-primitive-sequence (5),
 link-established (6),
 no-information (7) }

Access read-only

Status mandatory

Description The last link event which occurred for this port

Sequence ::= { fcEosPortEntry 150 }

fcEosPortLinkEventTime

Syntax DisplayString (SIZE (1..64))

Access read-only

Status mandatory

Description The time at which the last link event occurred for this port. If no link event has occurred this value shall be zero. The time is expressed as an ascii string, format TBD.

Sequence ::= { fcEosPortEntry 151 }

fcEosPortName

Syntax DisplayString (SIZE (0..24))

Access read-write

Status	mandatory
Description	A string describing the addressed port.
Sequence	::= { fcEosPortEntry 152 }

fcEosPortWWN

Syntax	FcEosPortWWN
Access	read-only
Status	mandatory
Description	The Port WWN.
Sequence	::= { fcEosPortEntry 153 }

NPIV Information

fcEosPortNPIVIndex

Syntax	VirtualPortNPIV
Access	not-accessible
Status	mandatory
Description	The NPIV-Index bit map that identifies the list of Virtual Ports associated with the FV_port. Only applicable for fVPorts.
Sequence	::= { fcEosPortEntry 154 }

fcEosPortNPIVMaxLogins

Syntax	INTEGER (1..256)
Access	not-accessible
Status	mandatory
Description	The maximum number of logins allowed for this port. To reduce fcEosPortNPIVMaxLogins below the number of devices that were currently logged in, then the port has to be offline.
Sequence	::= { fcEosPortEntry 155 }

Port Binding Table

fcEosPortBindingTable

Syntax	SEQUENCE OF FcEosPortBindingEntry
Access	not-accessible
Status	mandatory
Description	A table that contains one entry for each switch port.
Sequence	::= { fcEosPortBinding 1 }

fcEosPortBindingEntry

Syntax	FcEosPortBindingEntry
Access	not-accessible
Status	mandatory
Description	An entry containing the port binding information of the switch port. INDEX { fcEosPortBindingIndex }
Sequence	::= { fcEosPortBindingTable 1 } FcEosPortBindingEntry ::= SEQUENCE { fcEosPortBindingIndex FcEosPortIndex, fcEosPortBindingFlag INTEGER, fcEosPortConfiguredWWN FcEosPortWWN, fcEosPortAttachedWWN FcEosPortWWN }

fcEosPortBindingIndex

Syntax	FcEosPortIndex
Access	read-only
Status	mandatory
Description	The fixed physical port number on the switch. It ranges from 1 to the number of physical ports that can be supported in the switch.
Sequence	::= { fcEosPortBindingEntry 1 }

fcEosPortBindingFlag

Syntax	INTEGER { yes (1), no (2) }
Access	read-write
Status	mandatory
Description	The flag indicates whether or not Port Binding is in effect for an individual port.
Sequence	::= { fcEosPortBindingEntry 2 }

fcEosPortConfiguredWWN

Syntax	FcEosPortWWN
Access	read-write
Status	mandatory
Description	The authorized port WWN for attached servers and storage systems (F ports), or the authorized switch WWN for attached switches (E ports).
Sequence	::= { fcEosPortBindingEntry 3 }

fcEosPortAttachedWWN

Syntax	FcEosPortWWN
Access	read-only
Status	mandatory
Description	The WWN of the device currently attached to the port whether it has successfully connected or is currently being rejected due to a Port Binding violation.
Sequence	::= { fcEosPortBindingEntry 4 }

Fibre Channel Zoning Group

This group contains the current zoning configuration.

fcEosActiveZoneSetName

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	The active zone set name. This value will return NULL if the active zone set state is disabled
Sequence	::= { fcEosZoning 1 }

fcEosActiveZoneCount

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The number of zones in the active zone set. This value will return 0 if the active zone set state is disabled.
Sequence	::= { fcEosZoning 2 }

fcEosDefaultZoneSetState

Syntax	INTEGER{ enabled(1), disabled(2) }
Access	read-only
Status	mandatory
Description	The set state of the default zone set (1=Enabled,2= Disabled).
Sequence	::= { fcEosZoning 3 }

fcEosActiveZoneSetState

Syntax	INTEGER{ enabled(1), disabled(2) }
Access	read-only
Status	mandatory

Description	The state of the active zone set. If the active zone set state is disabled, then ActiveZoneSetName and ActiveZoneSetCount are invalid (1=Enabled,2= Disabled).
Sequence	::= { fcEosZoning 4 }

fcEosHardwareEnforcedZoning

Syntax	INTEGER{ yes (1), no (2) }
Access	read-only
Status	mandatory
Description	Indicates if zoning is hardware enforced (1=Yes, 2=No).
Sequence	::= { fcEosZoning 5 }

Active Zone Table

fcEosActiveZoneTable

Syntax	SEQUENCE OF FcEosActiveZoneEntry
Access	not-accessible
Status	mandatory
Description	A table that contains one entry for each zone in the active zone set.
Sequence	::= { fcEosZoning 6 }

fcEosActiveZoneEntry

Syntax	FcEosActiveZoneEntry
Access	not-accessible
Status	mandatory
Description	An entry containing the information specific to a zone.
	INDEX { fcEosZoneIndex }
Sequence	::= { fcEosActiveZoneTable 1 }

```

FcEosActiveZoneEntry ::= SEQUENCE {
    fcEosZoneIndex          INTEGER,
    fcEosZoneName           DisplayString,
    fcEosZoneMemberCount    INTEGER }

```

fcEosZoneIndex

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Zone index number. This number will range from 1 to the number of zones specified by the ActiveZoneCount.
Sequence	::= { fcEosActiveZoneEntry 1 }

fcEosZoneName

Syntax	DisplayString
Access	read-only
Status	mandatory
Description	This object is the name of this zone entry.
Sequence	::= { fcEosActiveZoneEntry 2 }

fcEosZoneMemberCount

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	This object contains a count of the number of members in this zone entry.
Sequence	::= { fcEosActiveZoneEntry 3 }

Active Member Table

fcEosActiveMemberTable

Syntax	SEQUENCE OF FcEosActiveMemberEntry
Access	not-accessible
Status	mandatory
Description	A table that contains one entry for each member in the active zone set.
Sequence	::= { fcEosZoning 7 }

fcEosActiveMemberEntry

Syntax	FcEosActiveMemberEntry
Access	not-accessible
Status	mandatory
Description	An entry containing the information specific to a member. INDEX { fcEosMemberZoneIndex, fcEosMemberIndex }
Sequence	::= { fcEosActiveMemberTable 1 } FcEosActiveMemberEntry ::= SEQUENCE { fcEosMemberZoneIndex INTEGER, fcEosMemberIndex INTEGER, fcEosMemberType INTEGER, fcEosMemberWWN FcEosPortWWN, fcEosMemberDomainID INTEGER, fcEosMemberPortNumber INTEGER }

fcEosMemberZoneIndex

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The index of the zone that this member belongs to. This is the same value as fcEosZoneIndex.

Sequence	::= { fcEosActiveMemberEntry 1 }
<hr/>	
fcEosMemberIndex	
Syntax	INTEGER
Access	read-only
Status	mandatory
Description	Member index number. This number will range from 1 to the number of members specified by the corresponding ZoneMemberCount.
Sequence	::= { fcEosActiveMemberEntry 2 }
<hr/>	
fcEosMemberType	
Syntax	INTEGER { wwn(1), portnumber(2) }
Access	read-only
Status	mandatory
Description	This object is the type of addressing that is associated with this member.
Sequence	::= { fcEosActiveMemberEntry 3 }
<hr/>	
fcEosMemberWWN	
Syntax	FcEosPortWWN
Access	read-only
Status	mandatory
Description	The WWN name as an 8-octet string. This value is only valid if the member type is 1, otherwise it will return all zeros.
Sequence	::= { fcEosActiveMemberEntry 4 }
<hr/>	
fcEosMemberDomainID	
Syntax	INTEGER
Access	read-only
Status	mandatory

Description The domain ID. This value is only valid if the member type is 2, otherwise it will return NULL.

Sequence ::= { fcEosActiveMemberEntry 5 }

fcEosMemberPortNumber

Syntax INTEGER

Access read-only

Status mandatory

Description The port number. This value is only valid if the member type is 2, otherwise it will return NULL.

Sequence ::= { fcEosActiveMemberEntry 6 }

Fibre Channel Threshold Alert Group

This group contains the threshold alert configuration.

Threshold Alert Table

fcEosTATable

Syntax SEQUENCE OF FcEosTAEntry

Access not-accessible

Status mandatory

Description A table that contains one entry for each configured threshold alert.

Sequence ::= { fcEosTA 1 }

fcEosTAEntry

Syntax FcEosTAEntry

Access not-accessible

Status mandatory

Description An entry containing a threshold alert configuration.

INDEX { fcEosTAIndex }

Sequence ::= { fcEosTATable 1 }

```

FcEosTAEEntry ::= SEQUENCE {
    fcEosTAIndex          INTEGER,
    fcEosTAName           DisplayString,
    fcEosTASState         INTEGER,
    fcEosTAType           INTEGER,
    fcEosTAPortType       INTEGER,
    fcEosTAPortList       FcEosPortList,
    fcEosTAInterval       INTEGER,
    fcEosTATriggerValue   INTEGER,
    fcEosTTADirection     INTEGER,
    fcEosTTATriggerDuration INTEGER,
    fcEosCTACounter       INTEGER }

```

fcEosTAIndex

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	This object is used to identify which threshold has been triggered.
Sequence	::= { fcEosTAEEntry 1 }

fcEosTAName

Syntax	DisplayString (SIZE (1..64))
Access	read-only
Status	mandatory
Description	The threshold alert name.
Sequence	::= { fcEosTAEEntry 2 }

fcEosTASState

Syntax	INTEGER { enabled (1), disabled (2) }
Access	read-only

Status	mandatory
Description	The current state of the threshold.
Sequence	::= { fcEosTAEEntry 3 }

fcEosTAType

Syntax	INTEGER { throughput (1), counter (2) }
Access	read-only
Status	mandatory
Description	The type of the threshold.
Sequence	::= { fcEosTAEEntry 4 }

fcEosTAPortType

Syntax	INTEGER { list (1), ePorts (2), fPorts (3), flPorts (4) }
Access	read-only
Status	mandatory
Description	A threshold can be set on a list of physical port numbers or on all the ports of the specified type (ePorts, fPorts).
Sequence	::= { fcEosTAEEntry 5 }

fcEosTAPortList

Syntax	FcEosPortList
Access	read-only
Status	mandatory
Description	A bit map that identifies which ports this threshold alert applies to (only valid if the Threshold Alert Port Type = list). The left most bit represents the port 0.
Sequence	::= { fcEosTAEEntry 6 }

fcEosTAInterval

Syntax	INTEGER
Access	read-only

Status	mandatory
Description	The number of minutes in a threshold alert interval.
Sequence	::= { fcEosTAEntry 7 }
<hr/>	
fcEosTATriggerValue	
Syntax	INTEGER
Access	read-only
Status	mandatory
Description	If the alert type is a Throughput Threshold Alert, then this is the percent utilization (1-100) required to trigger an alert. If the alert type is a Counter Threshold Alert, then this is the counter delta required to trigger an alert.
Sequence	::= { fcEosTAEntry 8 }
<hr/>	
fcEosTTADirection	
Syntax	INTEGER { transmit (1), receive (2), either (3) -- (Tx or Rx) }
Access	read-only
Status	mandatory
Description	This only applies when the alert type is a Throughput Threshold Alert. It specifies the throughput direction of the threshold.
Sequence	::= { fcEosTAEntry 9 }
<hr/>	
fcEosTTATriggerDuration	
Syntax	INTEGER
Access	read-only
Status	mandatory
Description	This only applies when the alert type is a Throughput Threshold Alert. It specifies the amount of time during a threshold alert interval that the trigger must be exceeded before an alert is generated.
Sequence	::= { fcEosTAEntry 10 }

fcEosCTACounter

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	This only applies when the alert type is a Counter Threshold Alert. It specifies statistical counter or counter set to be monitored.
Sequence	::= { fcEosTAEntry 11 }

FCEOS Enterprise-specific Trap Definitions

fcEosPortScn

Enterprise	mcData
Variables	{ fcEosPortOpStatus }
Description	An fcEosPortScn(1) is generated whenever a Fc_Port changes its operational state. For instance, the Fc_Port goes from on-line to offline.
Sequence	::= 1

fcEosFruScn

Enterprise	mcData
Variables	{ fcEosFruStatus }
Description	An fcEosFruScn(2) is generated whenever a FRU status changes its operational state.
Sequence	::= 2

fcEosPortBindingViolation

Enterprise	mcData
Variables	{ fcEosPortAttachedWWN }
Description	An fcEosPortBindingViolation(3) is generated whenever the switch detects that a port binding violation occurs.
Sequence	::= 3

fcEosThresholdAlert

Enterprise	mcData
Variables	{ fcEosPortIndex, fcEosTAIndex }
Description	An fcEosThresholdAlert(4) is generated whenever a threshold alert occurs.
Sequence	::= 4

New traps added for EOS 6.0:

These are intended to make it easier to interface the switch traps with management applications.

fcEosFruRemoved

Enterprise	mcData
Variables	{ fcEosFruCode, fcEosFruPosition, fcEosSysSwitchName, fcEosSysSwitchId }
Description	A fcEosFruRemoved trap is generated whenever a FRU is removed or transitions to an unknown status.
Sequence	::= 5

fcEosFruActive

Enterprise	mcData
Variables	{ fcEosFruCode, fcEosFruPosition, fcEosSysSwitchName, fcEosSysSwitchId }
Description	A fcEosFruActive trap is generated whenever a FRU transitions to an active status.
Sequence	::= 6

fcEosFruBackup

Enterprise	mcData
Variables	{ fcEosFruCode, fcEosFruPosition, fcEosSysSwitchName, fcEosSysSwitchId }
Description	A fcEosFruBackup trap is generated whenever a FRU transitions to a backup status.
Sequence	::= 7

fcEosFruUpdate

Enterprise	mcData
Variables	{ fcEosFruCode, fcEosFruPosition, fcEosSysSwitchName, fcEosSysSwitchId }

Description A fcEosFruUpdate trap is generated whenever a FRU transitions to an update/busy status.

Sequence ::= 8

fcEosFruFailed

Enterprise mcData

Variables { fcEosFruCode, fcEosFruPosition, fcEosSysSwitchName, fcEosSysSwitchId }

Description A fcEosFruFailed trap is generated whenever a FRU transitions to a failed status.

Sequence ::= 9

fcEosLinkBitErrorEvent

Enterprise mcData

Variables { fcEosPortIndex, fcEosPortName, fcEosPortWWN, fcEosSysSwitchName, fcEosSysSwitchId }

Description A fcEosLinkBitErrorEvent trap is generated when the bit error rate for a link exceeds an allowed threshold.

Sequence ::= 10

fcEosLinkNoSignalEvent

Enterprise mcData

Variables { fcEosPortIndex, fcEosPortName, fcEosPortWWN, fcEosSysSwitchName, fcEosSysSwitchId }

Description A fcEosLinkNoSignalEvent trap is generated when there is a loss of signal or sync.

Sequence ::= 11

fcEosLinkNOSEvent

Enterprise mcData

Variables { fcEosPortIndex, fcEosPortName, fcEosPortWWN, fcEosSysSwitchName, fcEosSysSwitchId }

Description A fcEosLinkNOSEvent trap is generated when a not operational primitive sequence is received.

Sequence ::= 12

fcEosLinkFailureEvent

Enterprise mcData

Variables { fcEosPortIndex, fcEosPortName, fcEosPortWWN, fcEosSysSwitchName, fcEosSysSwitchId }

Description A fcEosLinkFailureEvent trap is generated when a primitive sequence timeout occurs.

Sequence ::= 13

fcEosLinkInvalidEvent

Enterprise mcData

Variables { fcEosPortIndex, fcEosPortName, fcEosPortWWN, fcEosSysSwitchName, fcEosSysSwitchId }

Description A fcEosLinkInvalidEvent trap is generated when an invalid primitive sequence is detected.

Sequence ::= 14

fcEosLinkAddedEvent

Enterprise mcData

Variables { fcEosPortIndex, fcEosPortName, fcEosPortWWN, fcEosSysSwitchName, fcEosSysSwitchId }

Description A fcEosLinkAddedEvent trap is generated when the firmware detects that a new connection has been established on a port.

Sequence ::= 15

SNMP Framework MIB

Extracted from RFC 2271

October 1998, Ramanathan R. Kavasseri

Copyright (c) 1998 by cisco Systems, Inc.

All rights reserved.

This mib was extracted from RFC 2271.

SNMP-FRAMEWORK-MIB DEFINITIONS ::= BEGIN

IMPORTS

MODULE-IDENTITY, OBJECT-TYPE, OBJECT-IDENTITY,
snmpModules

FROM SNMPv2-SMI

TEXTUAL-CONVENTION

FROM SNMPv2-TC

MODULE-COMPLIANCE, OBJECT-GROUP

FROM SNMPv2-CONF;

snmpFrameworkMIB

Last-Updated 9709300000Z -- 30 September 1997

Organization SNMPv3 Working Group

Description	The SNMP Management Architecture MIB.
Sequence	::= { snmpModules 10 }

Textual Conventions used in the SNMP Management Architecture

SnmEngineID

Status	current
Description	<p>An SNMP engine's administratively-unique identifier.</p> <p>The value for this object may not be all zeros or all 'ff'H or the empty (zero length) string. The initial value for this object may be configured via an operator console entry or via an algorithmic function. In the latter case, the following example algorithm is recommended.</p> <p>In cases where there are multiple engines on the same system, the use of this algorithm is NOT appropriate, as it would result in all of those engines ending up with the same ID value.</p> <ol style="list-style-type: none"> 1. The very first bit is used to indicate how the rest of the data is composed. <ul style="list-style-type: none"> 0 as defined by enterprise using former methods that existed before SNMPv3. See item 2 below. 1 as defined by this architecture, see item 3 below. Note that this allows existing uses of the engineID (also known as AgentID [RFC1910]) to co-exist with any new uses. 2. The snmpEngineID has a length of 12 octets. <p>The first four octets are set to the binary equivalent of the agent's SNMP management private enterprise number as assigned by the Internet Assigned Numbers Authority (IANA). For example, if Acme Networks has been assigned { enterprises 696 }, the first four octets would be assigned '000002b8'H.</p> <p>The remaining eight octets are determined via one or more enterprise-specific methods. Such methods must be designed so as to maximize the possibility that the value of this object will be unique in the agent's administrative domain. For example, it may be the IP address of the SNMP entity, or the MAC address of one of the interfaces, with each address suitably padded with random</p>

octets. If multiple methods are defined, then it is recommended that the first octet indicate the method being used and the remaining octets be a function of the method.

- 3) The length of the octet strings varies. The first four octets are set to the binary equivalent of the agent's SNMP management private enterprise number as assigned by the Internet Assigned Numbers Authority (IANA).

For example, if Acme Networks has been assigned { enterprises 696 }, the first four octets would be assigned '000002b8'H. The very first bit is set to 1. For example, the above value for Acme Networks now changes to be '800002b8'H. The fifth octet indicates how the rest (6th and following octets) are formatted. The values for the fifth octet are:

- | | |
|---------|--|
| 0 | reserved, unused. |
| 1 | IPv4 address (4 octets) lowest non-special IP address. |
| 2 | IPv6 address (16 octets) lowest non-special IP address. |
| 3 | MAC address (6 octets) lowest IEEE MAC address, canonical order. |
| 4 | Text, administratively assigned Maximum remaining length 27. |
| 5 | Octets, administratively assigned. Maximum remaining length 27. |
| 6-127 | reserved, unused. |
| 127-255 | as defined by the enterprise. Maximum remaining length 27. |

SnmSecurityModel

Syntax	OCTET STRING (SIZE(1..32))
Status	current
Description	<p>An identifier that uniquely identifies a securityModel of the Security Subsystem within the SNMP Management Architecture. The values for securityModel are allocated as follows:</p> <ul style="list-style-type: none"> The zero value is reserved. Values between 1 and 255, inclusive, are reserved for standards-track Security Models and are managed by the Internet Assigned Numbers Authority (IANA).

- Values greater than 255 are allocated to enterprise-specific Security Models. An enterprise-specific securityModel value is defined to be: $\text{enterpriseID} * 256 + \text{security model within enterprise}$.

For example, the fourth Security Model defined by the enterprise whose enterpriseID is 1 would be 260. The eight bits allow a maximum of 255 (256-1 reserved) standards based Security Models. Similarly, they allow a maximum of 255 Security Models per enterprise.

It is believed that the assignment of new securityModel values will be rare in practice because the larger the number of simultaneously utilized Security Models, the larger the chance that interoperability will suffer. Consequently, it is believed that such a range will be sufficient. In the unlikely event that the standards committee finds this number to be insufficient over time, an enterprise number can be allocated to obtain an additional 255 possible values.

Note that the most significant bit must be zero; hence, there are 23 bits allocated for various Organizations to design and define non-standard securityModels. This limits the ability to define new proprietary implementations of Security Models to the first 8,388,608 enterprises.

It is worthwhile to note that, in its encoded form, the securityModel value will normally require only a single byte since, in practice, the leftmost bits will be zero for most messages and sign extension is suppressed by the encoding rules.

As of this writing, there are several values of securityModel defined for use with SNMP or reserved for use with supporting MIB Objects. They are as follows:

- | | |
|---|---------------------------------|
| 0 | reserved for 'any' |
| 1 | reserved for SNMPv1 |
| 2 | reserved for SNMPv2c |
| 3 | User-Based Security Model (USM) |

SnmpMessageProcessingModel

Syntax	INTEGER(0..2147483647)
Status	current

Description An identifier that uniquely identifies a Message Processing Model of the Message Processing Subsystem within a SNMP Management Architecture.

The values for `messageProcessingModel` are allocated as follows:

- Values between 0 and 255, inclusive, are reserved for standards-track Message Processing Models and are managed by the Internet Assigned Numbers Authority (IANA).
- Values greater than 255 are allocated to enterprise-specific Message Processing Models. An enterprise `messageProcessingModel` value is defined to be: $\text{enterpriseID} * 256 + \text{messageProcessingModel}$ within enterprise

For example, the fourth Message Processing Model defined by the enterprise whose `enterpriseID` is 1 would be 260. The eight bits allow a maximum of 256 standards based Message Processing Models. Similarly, they allow a maximum 256 Message Processing Models per enterprise.

It is believed that the assignment of new `messageProcessingModel` values will be rare in practice because the larger the number of simultaneously utilized Message Processing Models, the larger the chance that interoperability will suffer. It is believed that such a range will be sufficient. In the unlikely event that the standards committee finds this number to be insufficient over time, an enterprise number can be allocated to obtain an additional 256 possible values.

Note that the most significant bit must be zero; hence, there are 23 bits allocated for various Organizations to design and define non-standard `messageProcessingModels`. This limits the ability to define new proprietary implementations of Message Processing Models to the first 8,388,608 enterprises.

It is worthwhile to note that, in its encoded form, the `securityModel` value will normally require only a single byte since, in practice, the leftmost bits will be zero for most messages and sign extension is suppressed by the encoding rules.

As of this writing, there are several values of `messageProcessingModel` defined for use with SNMP. They are as follows:

0 reserved for SNMPv1

1	reserved for SNMPv2c
2	reserved for SNMPv2u and SNMPv2*
3	reserved for SNMPv3

SnmpSecurityLevel

Syntax	INTEGER(0..2147483647)
Status	current
Description	<p>A Level of Security at which SNMP messages can be sent or with which operations are being processed; in particular, one of:</p> <p>noAuthNoPriv without authentication and without privacy, authNoPriv with authentication but without privacy, authPriv with authentication and with privacy.</p> <p>These three values are ordered such that noAuthNoPriv is less than authNoPriv and authNoPriv is less than authPriv.</p>

SnmpAdminString

Syntax	<p>INTEGER { noAuthNoPriv(1), authNoPriv(2), authPriv(3) }</p> <p>DISPLAY-HINT "255a"</p>
Status	current
Description	<p>An octet string containing administrative information, preferably in human-readable form. To facilitate internationalization, this information is represented using the ISO/IEC IS 10646-1 character set, encoded as an octet string using the UTF-8 transformation format described in [RFC2044].</p> <p>Since additional code points are added by amendments to the 10646 standard from time to time, implementations must be prepared to encounter any code point from 0x00000000 to 0x7fffffff.</p> <p>The use of control codes should be avoided. When it is necessary to represent a newline, the control code sequence CR LF should be used. The use of leading or trailing white space should be avoided.</p> <p>For code points not directly supported by user interface hardware or software, an alternative means of entry and display, such as hexadecimal, may be provided.</p>

For information encoded in 7-bit US-ASCII, the UTF-8 encoding is identical to the US-ASCII encoding.

Note that when this TC is used for an object that is used or envisioned to be used as an index, then a SIZE restriction must be specified so that the number sub-identifiers for any object instance do not exceed the limit of 128, as defined by [RFC1905].

Syntax OCTET STRING (SIZE (0..255))

Administrative assignments

snmpFrameworkAdmin	OBJECT IDENTIFIER	::= { snmpFrameworkMIB 1 }
snmpFrameworkMIBObjects	OBJECT IDENTIFIER	::= { snmpFrameworkMIB 2 }
snmpFrameworkMIBConformance	OBJECT IDENTIFIER	::= { snmpFrameworkMIB 3 }

The snmpEngine Group

snmpEngine	OBJECT IDENTIFIER	::= { snmpFrameworkMIBObjects 1 }
------------	-------------------	-----------------------------------

snmpEngineID

Syntax	SnmpEngineID
Max-Access	read-only
Status	current
Description	An SNMP engine's administratively-unique identifier.
Sequence	::= { snmpEngine 1 }

snmpEngineBoots

Syntax	INTEGER (1..2147483647)
Max-Access	read-only
Status	current

Description	The number of times that the SNMP engine has (re-)initialized itself since its initial configuration.
-------------	---

Sequence	::= { snmpEngine 2 }
----------	----------------------

snmpEngineTime

Syntax	INTEGER (0..2147483647)
--------	-------------------------

Max-Access	read-only
------------	-----------

Status	current
--------	---------

Description	The number of seconds since the SNMP engine last incremented the snmpEngineBoots object.
-------------	--

Sequence	::= { snmpEngine 3 }
----------	----------------------

snmpEngineMaxMessageSize

Syntax	INTEGER (484..2147483647)
--------	---------------------------

Max-Access	read-only
------------	-----------

Status	current
--------	---------

Description	The maximum length in octets of an SNMP message which this SNMP engine can send or receive and process, determined as the minimum of the maximum message size values supported among all of the transports available to and supported by the engine.
-------------	--

Sequence	::= { snmpEngine 4 }
----------	----------------------

Registration Points for Authentication and Privacy Protocols

snmpAuthProtocols

Status	current
--------	---------

Description	Registration point for standards-track authentication protocols used in SNMP Management Frameworks.
-------------	---

Sequence	::= { snmpFrameworkAdmin 1 }
----------	------------------------------

snmpPrivProtocols

Status	current
--------	---------

Description Registration point for standards-track privacy protocols used in SNMP Management Frameworks.

Sequence ::= { snmpFrameworkAdmin 2 }

Conformance information

snmpFrameworkMI OBJECT ::=
BCompliances IDENTIFIER {snmpFrameworkMIBConformance 1}

snmpFrameworkMI OBJECT ::=
BGroups IDENTIFIER {snmpFrameworkMIBConformance 2}

Compliance statements

snmpFrameworkMIBCompliance

Status current

Description The compliance statement for SNMP engines which implement the SNMP Management Framework MIB.

MODULE -- this module

MANDATORY-GROUPS { snmpEngineGroup }

Sequence ::= { snmpFrameworkMIBCompliances 1 }

Units of conformance

snmpEngineGroup

Objects { snmpEngineID, snmpEngineBoots, snmpEngineTime, snmpEngineMaxMessageSize }

Status current

Description A collection of Objects for identifying and determining the configuration and current timeliness values of an SNMP engine.

Sequence ::= { snmpFrameworkMIBGroups 1 }

Groups in MIB II

Removed EGP group (lwx)

Rename the MIB from rfc1213.mib to mib2.mib

May 14, 2002 lwx

RFC1213-MIB DEFINITIONS ::= BEGIN

IMPORTS

mgmt, NetworkAddress, IpAddress, Counter, Gauge, TimeTicks

FROM RFC1155-SMI

OBJECT-TYPE

FROM RFC-1212;

This MIB module uses the extended OBJECT-TYPE macro as defined in [14]; MIB-II (same prefix as MIB-I)

mib-2 OBJECT IDENTIFIER ::= { mgmt 1 }

Textual conventions

DisplayString ::= OCTET STRING

This data type is used to model textual information taken from the NVT ASCII character set. By convention, objects with this syntax are declared as having SIZE (0..255)

PhysAddress ::= OCTET STRING

This data type is used to model media addresses. For many types of media, this will be in a binary representation. For example, an ethernet address would be represented as a string of 6 octets.

Groups in MIB-II

system	OBJECT IDENTIFIER	::= { mib-2 1 }
interfaces	OBJECT IDENTIFIER	::= { mib-2 2 }
at	OBJECT IDENTIFIER	::= { mib-2 3 }
ip	OBJECT IDENTIFIER	::= { mib-2 4 }
icmp	OBJECT IDENTIFIER	::= { mib-2 5 }
tcp	OBJECT IDENTIFIER	::= { mib-2 6 }
udp	OBJECT IDENTIFIER	::= { mib-2 7 }
egp	OBJECT IDENTIFIER	::= { mib-2 8 }

Historical (some say hysterical)

cmot	OBJECT IDENTIFIER ::= { mib-2 9 }
transmission	OBJECT IDENTIFIER ::= { mib-2 10 }
snmp	OBJECT IDENTIFIER ::= { mib-2 11 }

System group

Implementation of the System group is mandatory for all systems. If an agent is not configured to have a value for any of these variables, a string of length 0 is returned.

sysDescr

Syntax	DisplayString (SIZE (0..255))
Access	read-only
Status	mandatory
Description	A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. It is mandatory that this only contain printable ASCII characters.
Sequence	::= { system 1 }

sysObjectID

Syntax	OBJECT IDENTIFIER
Access	read-only
Status	mandatory
Description	The vendor's authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining `what kind of box' is being managed. For example, if vendor `Flintstones, Inc.' was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier .3.6.1.4.1.4242.1.1 to its `Fred Router'.
Sequence	::= { system 2 }

sysUpTime

Syntax	TimeTicks
Access	read-only
Status	mandatory
Description	The time (in hundredths of a second) since the network management portion of the system was last re-initialized.
Sequence	::= { system 3 }

sysContact

Syntax	DisplayString (SIZE (0..255))
Access	read-write
Status	mandatory
Description	The textual identification of the contact person for this managed node, together with information on how to contact this person.
Sequence	::= { system 4 }

sysName

Syntax	DisplayString (SIZE (0..255))
Access	read-write

Status	mandatory
Description	An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name.
Sequence	::= { system 5 }

sysLocation

Syntax	DisplayString (SIZE (0..255))
Access	read-write
Status	mandatory
Description	The physical location of this node (e.g., 'telephone closet, 3rd floor').
Sequence	::= { system 6 }

sysServices

Syntax	INTEGER (0..127)
Access	read-only
Status	mandatory
Description	<p>A value which indicates the set of services that this entity primarily offers. The value is a sum. This sum initially takes the value zero, Then, for each layer, L, in the range 1 through 7, that this node performs transactions for, 2 raised to (L - 1) is added to the sum.</p> <p>For example, a node which performs primarily routing functions would have a value of 4 ($2^{(3-1)}$). In contrast, a node which is a host offering application services would have a value of 72 ($2^{(4-1)} + 2^{(7-1)}$). Note that in the context of the Internet suite of protocols, values should be calculated accordingly:</p>

Layer functionality

- 1 physical (e.g., repeaters)
- 2 datalink/subnetwork (e.g., bridges)
- 3 internet (e.g., IP gateways)
- 4 end-to-end (e.g., IP hosts)
- 7 applications (e.g., mail relays)

For systems including OSI protocols, layers 5 and 6 may also be counted.

Sequence ::= { system 7 }

Interfaces group

Implementation of the Interfaces group is mandatory for all systems.

ifNumber

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The number of network interfaces (regardless of their current state) present on this system.
Sequence	::= { interfaces 1 }

Interfaces table

The Interfaces table contains information on the entity's interfaces. Each interface is thought of as being attached to a 'subnetwork'. Note that this term should not be confused with 'subnet' which refers to an addressing partitioning scheme used in the Internet suite of protocols.

ifTable

Syntax	SEQUENCE OF IfEntry
Access	not-accessible
Status	mandatory
Description	A list of interface entries. The number of entries is given by the value of ifNumber.
Sequence	::= { interfaces 2 }

ifEntry

Syntax	IfEntry
Access	not-accessible
Status	mandatory

Description An interface entry containing objects at the subnetwork layer and below for a particular interface.

INDEX { ifIndex }

Sequence ::= { ifTable 1 }

IfEntry ::=

SEQUENCE {

ifIndex INTEGER,

ifDescr DisplayString,

ifType INTEGER,

ifMtu INTEGER,

ifSpeed Gauge,

ifPhysAddress PhysAddress,

ifAdminStatus INTEGER,

ifOperStatus INTEGER,

ifLastChange TimeTicks,

ifInOctets Counter,

ifInUcastPkts Counter,

ifInNUcastPkts Counter,

ifInDiscards Counter,

ifInErrors Counter,

ifInUnknownProtos Counter,

ifOutOctets Counter,

ifOutUcastPkts Counter,

ifOutNUcastPkts Counter,

ifOutDiscards Counter,

ifOutErrors Counter,

ifOutQLen Gauge,

ifSpecific OBJECT IDENTIFIER }

ifIndex

Syntax INTEGER

Access read-only

Status	mandatory
Description	A unique value for each interface. Its value ranges between 1 and the value of ifNumber. The value for each interface must remain constant at least from one re-initialization of the entity's network management system to the next re- initialization.
Sequence	::= { ifEntry 1 }

ifDescr

Syntax	DisplayString (SIZE (0..255))
Access	read-only
Status	mandatory
Description	A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the hardware interface.
Sequence	::= { ifEntry 2 }

ifType

Syntax	INTEGER { other(1), -- none of the following regular1822(2), hdl1822(3), ddn-x25(4), rfc877-x25(5), ethernet-csmacd(6), iso88023-csmacd(7), iso88024-tokenBus(8), iso88025-tokenRing(9), iso88026-man(10), starLan(11), proteon-10Mbit(12), proteon-80Mbit(13),
--------	---

```

hyperchannel(14),
fddi(15),
lapb(16),
sdlc(17),
ds1(18),      -- T-1
e1(19),       -- european equiv. of T-1
basicISDN(20),
primaryISDN(21), -- proprietary serial
propPointToPointSerial(22),
ppp(23),
softwareLoopback(24),
eon(25),      -- CLNP over IP [11]
ethernet-3Mbit(26),
nsip(27),     -- XNS over IP
slip(28),     -- generic SLIP
ultra(29),    -- ULTRA technologies
ds3(30),     -- T-3
sip(31),     -- SMDS
frame-relay(32) }

```

Access	read-only
Status	mandatory
Description	The type of interface, distinguished according to the physical/link protocol(s) immediately 'below' the network layer in the protocol stack.
Sequence	::= { ifEntry 3 }

ifMtu

Syntax	INTEGER
Access	read-only
Status	mandatory

Description The size of the largest datagram which can be sent/received on the interface, specified in octets. For interfaces that are used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface.

Sequence ::= { ifEntry 4 }

ifSpeed

Syntax Gauge

Access read-only

Status mandatory

Description An estimate of the interface's current bandwidth in bits per second. For interfaces which do not vary in bandwidth or for those where no accurate estimation can be made, this object should contain the nominal bandwidth.

Sequence ::= { ifEntry 5 }

ifPhysAddress

Syntax PhysAddress

Access read-only

Status mandatory

Description The interface's address at the protocol layer immediately 'below' the network layer in the protocol stack. For interfaces which do not have such an address (e.g., a serial line), this object should contain an octet string of zero length.

Sequence ::= { ifEntry 6 }

ifAdminStatus

Syntax INTEGER {
 up(1), -- ready to pass packets
 down(2),
 testing(3) -- in some test mode }

Access read-write

Status mandatory

Description The desired state of the interface. The testing(3) state indicates that no operational packets can be passed.

Sequence ::= { ifEntry 7 }

ifOperStatus

Syntax INTEGER {
 up(1), -- ready to pass packets
 down(2),
 testing(3) -- in some test mode }

Access read-only

Status mandatory

Description The current operational state of the interface. The testing(3) state indicates that no operational packets can be passed.

Sequence ::= { ifEntry 8 }

ifLastChange

Syntax TimeTicks

Access read-only

Status mandatory

Description The value of sysUpTime at the time the interface entered its current operational state. If the current state was entered prior to the last re-initialization of the local network management subsystem, then this object contains a zero value.

Sequence ::= { ifEntry 9 }

ifInOctets

Syntax Counter

Access read-only

Status mandatory

Description The total number of octets received on the interface, including framing characters.

Sequence ::= { ifEntry 10 }

ifInUcastPkts

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of subnetwork-unicast packets delivered to a higher-layer protocol.
Sequence	::= { ifEntry 11 }

ifInNUcastPkts

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of non-unicast (i.e., subnetwork- broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol.
Sequence	::= { ifEntry 12 }

ifInDiscards

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.
Sequence	::= { ifEntry 13 }

ifInErrors

Syntax	Counter
Access	read-only
Status	mandatory

Description The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.

Sequence ::= { ifEntry 14 }

ifInUnknownProtos

Syntax Counter

Access read-only

Status mandatory

Description The number of packets received via the interface which were discarded because of an unknown or unsupported protocol.

Sequence ::= { ifEntry 15 }

ifOutOctets

Syntax Counter

Access read-only

Status mandatory

Description The total number of octets transmitted out of the interface, including framing characters."

Sequence ::= { ifEntry 16 }

ifOutUcastPkts

Syntax Counter

Access read-only

Status mandatory

Description The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent.

Sequence ::= { ifEntry 17 }

ifOutNUcastPkts

Syntax Counter

Access	read-only
Status	mandatory
Description	The total number of packets that higher-level protocols requested be transmitted to a non-unicast (i.e., a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent.
Sequence	::= { ifEntry 18 }

ifOutDiscards

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.
Sequence	::= { ifEntry 19 }

ifOutErrors

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of outbound packets that could not be transmitted because of errors.
Sequence	::= { ifEntry 20 }

ifOutQLen

Syntax	Gauge
Access	read-only
Status	mandatory
Description	The length of the output packet queue (in packets).
Sequence	::= { ifEntry 21 }

ifSpecific

Syntax	OBJECT IDENTIFIER
Access	read-only
Status	mandatory
Description	A reference to MIB definitions specific to the particular media being used to realize the interface. For example, if the interface is realized by an ethernet, then the value of this object refers to a document defining objects specific to ethernet. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntatically valid object identifier, and any conformant implementation of ASN.1 and BER must be able to generate and recognize this value.
Sequence	::= { ifEntry 22 }

Address Translation group

Implementation of the Address Translation group is mandatory for all systems. Note however that this group is deprecated by MIB-II. That is, it is being included solely for compatibility with MIB-I nodes, and will most likely be excluded from MIB-III nodes. From MIB-II and onwards, each network protocol group contains its own address translation tables.

The Address Translation group contains one table which is the union across all interfaces of the translation tables for converting a NetworkAddress (e.g., an IP address) into a subnetwork-specific address. For lack of a better term, this document refers to such a subnetwork-specific address as a 'physical' address.

Examples of such translation tables are: for broadcast media where ARP is in use, the translation table is equivalent to the ARP cache; or, on an X.25 network where non-algorithmic translation to X.121 addresses is required, the translation table contains the NetworkAddress to X.121 address equivalences.

atTable

Syntax	SEQUENCE OF AtEntry
Access	not-accessible

Status	deprecated
Description	The Address Translation tables contain the NetworkAddress to 'physical' address equivalences. Some interfaces do not use translation tables for determining address equivalences (e.g., DDN-X.25 has an algorithmic method); if all interfaces are of this type, then the Address Translation table is empty, i.e., has zero entries.
Sequence	::= { at 1 }

atEntry

Syntax	AtEntry
Access	not-accessible
Status	deprecated
Description	Each entry contains one NetworkAddress to physical' address equivalence. INDEX { atIfIndex, atNetAddress }
Sequence	::= { atTable 1 } AtEntry ::= SEQUENCE { atIfIndex INTEGER, atPhysAddress PhysAddress, atNetAddress NetworkAddress }

atIfIndex

Syntax	INTEGER
Access	read-write
Status	deprecated
Description	The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.
Sequence	::= { atEntry 1 }

atPhysAddress

Syntax	PhysAddress
Access	read-write
Status	deprecated
Description	The media-dependent `physical' address. Setting this object to a null string (one of zero length) has the effect of invaliding the corresponding entry in the atTable object. That is, it effectively dissasociates the interface identified with said entry from the apping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant atPhysAddress object.
Sequence	::= { atEntry 2 }

atNetAddress

Syntax	NetworkAddress
Access	read-write
Status	deprecated
Description	The NetworkAddress (e.g., the IP address) corresponding to the media-dependent `physical' address.
Sequence	::= { atEntry 3 }

IP group

Implementation of the IP group is mandatory for all systems.

ipForwarding

Syntax	INTEGER { forwarding(1), -- acting as a gateway not-forwarding(2) -- NOT acting as a gateway }
Access	read-write

Status	mandatory
Description	<p>The indication of whether this entity is acting as an IP gateway in respect to the forwarding of datagrams received by, but not addressed to, this entity. IP gateways forward datagrams. IP hosts do not (except those source-routed via the host).</p> <p>Note that for some managed nodes, this object may take on only a subset of the values possible. Accordingly, it is appropriate for an agent to return a 'badValue' response if a management station attempts to change this object to an inappropriate value.</p>
Sequence	::= { ip 1 }

ipDefaultTTL

Syntax	INTEGER
Access	read-write
Status	mandatory
Description	<p>The default value inserted into the Time-To-Live field of the IP header of datagrams originated at this entity, whenever a TTL value is not supplied by the transport layer protocol.</p>
Sequence	::= { ip 2 }

ipInReceives

Syntax	Counter
Access	read-only
Status	mandatory
Description	<p>The total number of input datagrams received from interfaces, including those received in error.</p>
Sequence	::= { ip 3 }

ipInHdrErrors

Syntax	Counter
Access	read-only
Status	mandatory

Description The number of input datagrams discarded due to errors in their IP headers, including bad checksums, version number mismatch, other format errors, time-to-live exceeded, errors discovered in processing their IP options, etc.

Sequence ::= { ip 4 }

ipInAddrErrors

Syntax Counter

Access read-only

Status mandatory

Description The number of input datagrams discarded because the IP address in their IP header's destination field was not a valid address to be received at this entity. This count includes invalid addresses (e.g., 0.0.0.0) and addresses of unsupported Classes (e.g., Class E). For entities which are not IP Gateways and therefore do not forward datagrams, this counter includes datagrams discarded because the destination address was not a local address.

Sequence ::= { ip 5 }

ipForwDatagrams

Syntax Counter

Access read-only

Status mandatory

Description The number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to find a route to forward them to that final destination. In entities which do not act as IP Gateways, this counter will include only those packets which were Source-Routed via this entity, and the Source-Route option processing was successful.

Sequence ::= { ip 6 }

ipInUnknownProtos

Syntax Counter

Access read-only

Status	mandatory
Description	The number of locally-addressed datagrams received successfully but discarded because of an unknown or unsupported protocol.
Sequence	::= { ip 7 }

ipInDiscards

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of input IP datagrams for which no problems were encountered to prevent their continued processing, but which were discarded (e.g., for lack of buffer space). Note that this counter does not include any datagrams discarded while awaiting re-assembly.
Sequence	::= { ip 8 }

ipInDelivers

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of input datagrams successfully delivered to IP user-protocols (including ICMP).
Sequence	::= { ip 9 }

ipOutRequests

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of IP datagrams which local IP user-protocols (including ICMP) supplied to IP in requests for transmission. Note that this counter does not include any datagrams counted in ipForwDatagrams.
Sequence	::= { ip 10 }

ipOutDiscards

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of output IP datagrams for which no problem was encountered to prevent their transmission to their destination, but which were discarded (e.g., for lack of buffer space). Note that this counter would include datagrams counted in ipForwDatagrams if any such packets met this (discretionary) discard criterion.
Sequence	::= { ip 11 }

ipOutNoRoutes

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of IP datagrams discarded because no route could be found to transmit them to their destination. Note that this counter includes any packets counted in ipForwDatagrams which meet this 'no-route' criterion. Note that this includes any datagrams which a host cannot route because all of its default gateways are down.
Sequence	::= { ip 12 }

ipReasmTimeout

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The maximum number of seconds which received fragments are held while they are awaiting reassembly at this entity.
Sequence	::= { ip 13 }

ipReasmReqds

Syntax	Counter
--------	---------

Access	read-only
Status	mandatory
Description	The number of IP fragments received which needed to be reassembled at this entity."
Sequence	::= { ip 14 }

ipReasmOKs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of IP datagrams successfully re-assembled.
Sequence	::= { ip 15 }

ipReasmFails

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of failures detected by the IP re-assembly algorithm (for whatever reason: timed out, errors, etc). Note that this is not necessarily a count of discarded IP fragments since some algorithms (notably the algorithm in RFC 815) can lose track of the number of fragments by combining them as they are received.
Sequence	::= { ip 16 }

ipFragOKs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of IP datagrams that have been successfully fragmented at this entity.
Sequence	::= { ip 17 }

ipFragFails

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of IP datagrams that have been discarded because they needed to be fragmented at this entity but could not be, e.g., because their Don't Fragment flag was set."
Sequence	::= { ip 18 }

ipFragCreates

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of IP datagram fragments that have been generated as a result of fragmentation at this entity.
Sequence	::= { ip 19 }

IP address table The IP address table contains this entity's IP addressing information.

ipAddrTable

Syntax	SEQUENCE OF IpAddrEntry
Access	not-accessible
Status	mandatory
Description	The table of addressing information relevant to this entity's IP addresses.
Sequence	::= { ip 20 }

ipAddrEntry

Syntax	IpAddrEntry
Access	not-accessible
Status	mandatory

Description The addressing information for one of this entity's IP addresses.

INDEX { ipAdEntAddr }

Sequence ::= { ipAddrTable 1 }

IpAddrEntry ::=

SEQUENCE {

ipAdEntAddr IpAddress,

ipAdEntIfIndex INTEGER,

ipAdEntNetMask IpAddress,

ipAdEntBcastAddr INTEGER,

ipAdEntReasmMaxSize INTEGER (0..65535) }

ipAdEntAddr

Syntax IpAddress

Access read-only

Status mandatory

Description The IP address to which this entry's addressing information pertains.

Sequence ::= { ipAddrEntry 1 }

ipAdEntIfIndex

Syntax INTEGER

Access read-only

Status mandatory

Description The index value which uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

Sequence ::= { ipAddrEntry 2 }

ipAdEntNetMask

Syntax IpAddress

Access read-only

Status	mandatory
Description	The subnet mask associated with the IP address of this entry. The value of the mask is an IP address with all the network bits set to 1 and all the hosts bits set to 0.
Sequence	::= { ipAddrEntry 3 }

ipAdEntBcastAddr

Syntax	INTEGER
Access	read-only
Status	mandatory
Description	The value of the least-significant bit in the IP broadcast address used for sending datagrams on the (logical) interface associated with the IP address of this entry. For example, when the Internet standard all-ones broadcast address is used, the value will be 1. This value applies to both the subnet and network broadcasts addresses used by the entity on this (logical) interface.
Sequence	::= { ipAddrEntry 4 }

ipAdEntReasmMaxSize

Syntax	INTEGER (0..65535)
Access	read-only
Status	mandatory
Description	The size of the largest IP datagram which this entity can re-assemble from incoming IP fragmented datagrams received on this interface.
Sequence	::= { ipAddrEntry 5 }

IP routing table	The IP routing table contains an entry for each route presently known to this entity.
-------------------------	---

ipRouteTable

Syntax	SEQUENCE OF IpRouteEntry
Access	not-accessible
Status	mandatory

Description	This entity's IP Routing table.
Sequence	::= { ip 21 }

ipRouteEntry

Syntax	IpRouteEntry
Access	not-accessible
Status	mandatory
Description	A route to a particular destination. INDEX { ipRouteDest }
Sequence	::= { ipRouteTable 1 } IpRouteEntry ::= SEQUENCE { ipRouteDest IpAddress, ipRouteIfIndex INTEGER, ipRouteMetric1 INTEGER, ipRouteMetric2 INTEGER, ipRouteMetric3 INTEGER, ipRouteMetric4 INTEGER, ipRouteNextHop IpAddress, ipRouteType INTEGER, ipRouteProto INTEGER, ipRouteAge INTEGER, ipRouteMask IpAddress, ipRouteMetric5 INTEGER, ipRouteInfo OBJECT IDENTIFIER }

ipRouteDest

Syntax	IpAddress
Access	read-write
Status	mandatory

Description The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. Multiple routes to a single destination can appear in the table, but access to such multiple entries is dependent on the table-access mechanisms defined by the network management protocol in use.

Sequence ::= { ipRouteEntry 1 }

ipRouteIfIndex

Syntax INTEGER

Access read-write

Status mandatory

Description The index value which uniquely identifies the local interface through which the next hop of this route should be reached. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.

Sequence ::= { ipRouteEntry 2 }

ipRouteMetric1

Syntax INTEGER

Access read-write

Status mandatory

Description The primary routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

Sequence ::= { ipRouteEntry 3 }

ipRouteMetric2

Syntax INTEGER

Access read-write

Status mandatory

Description An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's

ipRouteProto value. If this metric is not used, its value should be set to -1.

Sequence ::= { ipRouteEntry 4 }

ipRouteMetric3

Syntax INTEGER

Access read-write

Status mandatory

Description An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

Sequence ::= { ipRouteEntry 5 }

ipRouteMetric4

Syntax INTEGER

Access read-write

Status mandatory

Description An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.

Sequence ::= { ipRouteEntry 6 }

ipRouteNextHop

Syntax IpAddress

Access read-write

Status mandatory

Description The IP address of the next hop of this route. (In the case of a route bound to an interface which is realized via a broadcast media, the value of this field is the agent's IP address on that interface.)

Sequence ::= { ipRouteEntry 7 }

ipRouteType

Syntax	INTEGER { other(1), -- none of the following invalid(2), -- an invalidated route -- route to directly direct(3), -- connected (sub-)network route to a non-local indirect(4) -- host/network/sub-network }
Access	read-write
Status	mandatory
Description	<p>The type of route. Note that the values direct(3) and indirect(4) refer to the notion of direct and indirect routing in the IP architecture.</p> <p>Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipRouteTable object. That is, it effectively dissasociates the destination identified with said entry from the route identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table.</p> <p>Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipRouteType object.</p>
Sequence	::= { ipRouteEntry 8 }

ipRouteProto

Syntax	INTEGER { other(1), -- none of the following non-protocol information, e.g., manually configured local(2), -- entries set via a network netmgmt(3), -- management protocol obtained via ICMP, icmp(4), -- e.g., Redirect the remaining values are all gateway routing protocols egp(5), ggp(6),
--------	--

```

hello(7),
rip(8),
is-is(9),
es-is(10),
ciscoIgrp(11),
bbnSpflgp(12),
ospf(13),
bgp(14) }

```

Access	read-only
Status	mandatory
Description	The routing mechanism via which this route was learned. Inclusion of values for gateway routing protocols is not intended to imply that hosts should support those protocols.
Sequence	::= { ipRouteEntry 9 }

ipRouteAge

Syntax	INTEGER
Access	read-write
Status	mandatory
Description	The number of seconds since this route was last updated or otherwise determined to be correct. Note that no semantics of 'too old' can be implied except through knowledge of the routing protocol by which the route was learned.
Sequence	::= { ipRouteEntry 10 }

ipRouteMask

Syntax	IpAddress
Access	read-write
Status	mandatory
Description	Indicate the mask to be logical-ANDed with the destination address before being compared to the value in the ipRouteDest field. For those systems that do not support arbitrary subnet masks, an agent

constructs the value of the ipRouteMask by determining whether the value of the correspondent ipRouteDest field belong to a class-A, B, or C network, and then using one of:

mask	network
255.0.0.0	class-A
255.255.0.0	class-B
255.255.255.0	class-C

If the value of the ipRouteDest is 0.0.0.0 (a default route), then the mask value is also 0.0.0.0. It should be noted that all IP routing subsystems implicitly use this mechanism.

Sequence ::= { ipRouteEntry 11 }

ipRouteMetric5

Syntax	INTEGER
Access	read-write
Status	mandatory
Description	An alternate routing metric for this route. The semantics of this metric are determined by the routing-protocol specified in the route's ipRouteProto value. If this metric is not used, its value should be set to -1.
Sequence	::= { ipRouteEntry 12 }

ipRouteInfo

Syntax	OBJECT IDENTIFIER
Access	read-only
Status	mandatory
Description	A reference to MIB definitions specific to the particular routing protocol which is responsible for this route, as determined by the value specified in the route's ipRouteProto value. If this information is not present, its value should be set to the OBJECT IDENTIFIER { 0 0 }, which is a syntatically valid object identifier, and any conformant implementation of ASN.1 and BER must be able to generate and recognize this value.
Sequence	::= { ipRouteEntry 13 }

IP Address Translation table

The IP address translation table contain the IpAddress to `physical' address equivalences. Some interfaces do not use translation tables for determining address equivalences (e.g., DDN-X.25 has an algorithmic method); if all interfaces are of this type, then the Address Translation table is empty, i.e., has zero entries.

ipNetToMediaTable

Syntax	SEQUENCE OF IpNetToMediaEntry
Access	not-accessible
Status	mandatory
Description	The IP Address Translation table used for mapping from IP addresses to physical addresses.
Sequence	::= { ip 22 }

ipNetToMediaEntry

Syntax	IpNetToMediaEntry
Access	not-accessible
Status	mandatory
Description	Each entry contains one IpAddress to `physical' address equivalence. INDEX { ipNetToMediaIfIndex, ipNetToMediaNetAddress }
Sequence	::= { ipNetToMediaTable 1 } IpNetToMediaEntry ::= SEQUENCE { ipNetToMediaIfIndex INTEGER, ipNetToMediaPhysAddress PhysAddress, ipNetToMediaNetAddress IpAddress, ipNetToMediaType INTEGER }

ipNetToMediaIfIndex

Syntax	INTEGER
--------	---------

Access	read-write
Status	mandatory
Description	The interface on which this entry's equivalence is effective. The interface identified by a particular value of this index is the same interface as identified by the same value of ifIndex.
Sequence	::= { ipNetToMediaEntry 1 }

ipNetToMediaPhysAddress

Syntax	PhysAddress
Access	read-write
Status	mandatory
Description	The media-dependent 'physical' address.
Sequence	::= { ipNetToMediaEntry 2 }

ipNetToMediaNetAddress

Syntax	IpAddress
Access	read-write
Status	mandatory
Description	The IpAddress corresponding to the media-dependent 'physical' address.
Sequence	::= { ipNetToMediaEntry 3 }

ipNetToMediaType

Syntax	INTEGER { other(1), -- none of the following invalid(2), -- an invalidated mapping dynamic(3), static(4) }
Access	read-write
Status	mandatory
Description	The type of mapping.

Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the ipNetToMediaTable. That is, it effectively dissasociates the interface identified with said entry from the mapping identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management stations must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant ipNetToMediaType object.

Sequence ::= { ipNetToMediaEntry 4 }

Additional IP objects

ipRoutingDiscards

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of routing entries which were chosen to be discarded even though they are valid. One possible reason for discarding such an entry could be to free-up buffer space for other routing entries.
Sequence	::= { ip 23 }

ICMP group

Implementation of the ICMP group is mandatory for all systems.

icmpInMsgs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of ICMP messages which the entity received. Note that this counter includes all those counted by icmpInErrors.
Sequence	::= { icmp 1 }

icmpInErrors

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP messages which the entity received but determined as having ICMP-specific errors (bad ICMP checksums, bad length, etc.).
Sequence	::= { icmp 2 }

icmpInDestUnreachs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Destination Unreachable messages received.
Sequence	::= { icmp 3 }

icmpInTimeExcds

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Time Exceeded messages received.
Sequence	::= { icmp 4 }

icmpInParmProbs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Parameter Problem messages received.
Sequence	::= { icmp 5 }

icmpInSrcQuenchs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Source Quench messages received.
Sequence	::= { icmp 6 }

icmpInRedirects

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Redirect messages received.
Sequence	::= { icmp 7 }

icmpInEchos

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Echo (request) messages received.
Sequence	::= { icmp 8 }

icmpInEchoReps

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Echo Reply messages received.
Sequence	::= { icmp 9 }

icmpInTimestamps

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Timestamp (request) messages received.
Sequence	::= { icmp 10 }

icmpInTimestampReps

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Timestamp Reply messages received.
Sequence	::= { icmp 11 }

icmpInAddrMasks

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Address Mask Request messages received.
Sequence	::= { icmp 12 }

icmpInAddrMaskReps

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Address Mask Reply messages received.
Sequence	::= { icmp 13 }

icmpOutMsgs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of ICMP messages which this entity attempted to send. Note that this counter includes all those counted by icmpOutErrors.
Sequence	::= { icmp 14 }

icmpOutErrors

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP messages which this entity did not send due to problems discovered within ICMP such as a lack of buffers. This value should not include errors discovered outside the ICMP layer such as the inability of IP to route the resultant datagram. In some implementations there may be no types of error which contribute to this counter's value.
Sequence	::= { icmp 15 }

icmpOutDestUnreachs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Destination Unreachable messages sent.
Sequence	::= { icmp 16 }

icmpOutTimeExcds

Syntax	Counter
Access	read-only

Status	mandatory
Description	The number of ICMP Time Exceeded messages sent.
Sequence	::= { icmp 17 }

icmpOutParmProbs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Parameter Problem messages sent.
Sequence	::= { icmp 18 }

icmpOutSrcQuenchs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Source Quench messages sent.
Sequence	::= { icmp 19 }

icmpOutRedirects

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Redirect messages sent. For a host, this object will always be zero, since hosts do not send redirects.
Sequence	::= { icmp 20 }

icmpOutEchos

Syntax	Counter
Access	read-only
Status	mandatory

Description	The number of ICMP Echo (request) messages sent.
Sequence	::= { icmp 21 }

icmpOutEchoReps

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Echo Reply messages sent.
Sequence	::= { icmp 22 }

icmpOutTimestamps

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Timestamp (request) messages sent.
Sequence	::= { icmp 23 }

icmpOutTimestampReps

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Timestamp Reply messages sent.
Sequence	::= { icmp 24 }

icmpOutAddrMasks

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Address Mask Request messages sent.
Sequence	::= { icmp 25 }

icmpOutAddrMaskReps

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of ICMP Address Mask Reply messages sent.
Sequence	::= { icmp 26 }

TCP group

Implementation of the TCP group is mandatory for all systems that implement the TCP.

Note that instances of object types that represent information about a particular TCP connection are transient; they persist only as long as the connection in question.

tcpRtoAlgorithm

Syntax	INTEGER { other(1), -- none of the following constant(2), -- a constant rto rsre(3), -- MIL-STD-1778, Appendix B vanj(4) -- Van Jacobson's algorithm [10] }
Access	read-only
Status	mandatory
Description	The algorithm used to determine the timeout value used for retransmitting unacknowledged octets.
Sequence	::= { tcp 1 }

tcpRtoMin

Syntax	INTEGER
Access	read-only
Status	mandatory

Description The minimum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the LBOUND quantity described in RFC 793.

Sequence ::= { tcp 2 }

tcpRtoMax

Syntax INTEGER

Access read-only

Status mandatory

Description The maximum value permitted by a TCP implementation for the retransmission timeout, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the retransmission timeout. In particular, when the timeout algorithm is rsre(3), an object of this type has the semantics of the UBOUND quantity described in RFC 793.

Sequence ::= { tcp 3 }

tcpMaxConn

Syntax INTEGER

Access read-only

Status mandatory

Description The limit on the total number of TCP connections the entity can support. In entities where the maximum number of connections is dynamic, this object should contain the value -1.

Sequence ::= { tcp 4 }

tcpActiveOpens

Syntax Counter

Access read-only

Status mandatory

Description The number of times TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state.

Sequence ::= { tcp 5 }

tcpPassiveOpens

Syntax Counter

Access read-only

Status mandatory

Description The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state.

Sequence ::= { tcp 6 }

tcpAttemptFails

Syntax Counter

Access read-only

Status mandatory

Description The number of times TCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state.

Sequence ::= { tcp 7 }

tcpEstabResets

Syntax Counter

Access read-only

Status mandatory

Description The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state.

Sequence ::= { tcp 8 }

tcpCurrEstab

Syntax	Gauge
Access	read-only
Status	mandatory
Description	The number of TCP connections for which the current state is either ESTABLISHED or CLOSE- WAIT.
Sequence	::= { tcp 9 }

tcpInSegs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of segments received, including those received in error. This count includes segments received on currently established connections.
Sequence	::= { tcp 10 }

tcpOutSegs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets.
Sequence	::= { tcp 11 }

tcpRetransSegs

Syntax	Counter
Access	read-only
Status	mandatory

Description The total number of segments retransmitted - that is, the number of TCP segments transmitted containing one or more previously transmitted octets.

Sequence ::= { tcp 12 }

TCP Connection table The TCP connection table contains information about this entity's existing TCP connections.

tcpConnTable

Syntax SEQUENCE OF TcpConnEntry

Access not-accessible

Status mandatory

Description A table containing TCP connection-specific information.

Sequence ::= { tcp 13 }

tcpConnEntry

Syntax TcpConnEntry

Access not-accessible

Status mandatory

Description Information about a particular current TCP connection. An object of this type is transient, in that it ceases to exist when (or soon after) the connection makes the transition to the CLOSED state.

INDEX { tcpConnLocalAddress,
tcpConnLocalPort,
tcpConnRemAddress,
tcpConnRemPort }

Sequence ::= { tcpConnTable 1 }

TcpConnEntry ::=

SEQUENCE {

tcpConnState INTEGER,

tcpConnLocalAddress IpAddress,

tcpConnLocalPort	INTEGER (0..65535),
tcpConnRemAddress	IpAddress,
tcpConnRemPort	INTEGER (0..65535) }

tcpConnState

Syntax	INTEGER { closed(1), listen(2), synSent(3), synReceived(4), established(5), finWait1(6), finWait2(7), closeWait(8), lastAck(9), closing(10), timeWait(11), deleteTCB(12) }
--------	--

Access	read-write
--------	------------

Status	mandatory
--------	-----------

Description	The state of this TCP connection. The only value which may be set by a management station is deleteTCB(12). Accordingly, it is appropriate for an agent to return a 'badValue' response if a management station attempts to set this object to any other value.
-------------	---

If a management station sets this object to the value deleteTCB(12), then this has the effect of deleting the TCB (as defined in RFC 793) of the corresponding connection on the managed node, resulting in immediate termination of the connection.

As an implementation-specific option, a RST segment may be sent from the managed node to the other TCP endpoint (note however that RST segments are not sent reliably).

Sequence	::= { tcpConnEntry 1 }
----------	------------------------

tcpConnLocalAddress

Syntax	IpAddress
Access	read-only
Status	mandatory
Description	The local IP address for this TCP connection. In the case of a connection in the listen state which is willing to accept connections for any IP interface associated with the node, the value 0.0.0.0 is used.
Sequence	::= { tcpConnEntry 2 }

tcpConnLocalPort

Syntax	INTEGER (0..65535)
Access	read-only
Status	mandatory
Description	The local port number for this TCP connection.
Sequence	::= { tcpConnEntry 3 }

tcpConnRemAddress

Syntax	IpAddress
Access	read-only
Status	mandatory
Description	The remote IP address for this TCP connection.
Sequence	::= { tcpConnEntry 4 }

tcpConnRemPort

Syntax	INTEGER (0..65535)
Access	read-only
Status	mandatory
Description	The remote port number for this TCP connection.
Sequence	::= { tcpConnEntry 5 }

Additional TCP objects

tcpInErrs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of segments received in error (e.g., bad TCP checksums).
Sequence	::= { tcp 14 }

tcpOutRsts

Syntax	Counter
Access	read-only
Status	mandatory
Description	The number of TCP segments sent containing the RST flag.
Sequence	::= { tcp 15 }
UDP group	Implementation of the UDP group is mandatory for all systems which implement the UDP.

udpInDatagrams

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of UDP datagrams delivered to UDP users.
Sequence	::= { udp 1 }

udpNoPorts

Syntax	Counter
Access	read-only
Status	mandatory

Description	The total number of received UDP datagrams for which there was no application at the destination port.
-------------	--

Sequence	::= { udp 2 }
----------	---------------

udpInErrors

Syntax	Counter
--------	---------

Access	read-only
--------	-----------

Status	mandatory
--------	-----------

Description	The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port.
-------------	---

Sequence	::= { udp 3 }
----------	---------------

udpOutDatagrams

Syntax	Counter
--------	---------

Access	read-only
--------	-----------

Status	mandatory
--------	-----------

Description	The total number of UDP datagrams sent from this entity.
-------------	--

Sequence	::= { udp 4 }
----------	---------------

UDP Listener table	The UDP listener table contains information about this entity's UDP end-points on which a local application is currently accepting datagrams.
---------------------------	---

udpTable

Syntax	SEQUENCE OF UdpEntry
--------	----------------------

Access	not-accessible
--------	----------------

Status	mandatory
--------	-----------

Description	A table containing UDP listener information.
-------------	--

Sequence	::= { udp 5 }
----------	---------------

udpEntry

Syntax	UdpEntry
Access	not-accessible
Status	mandatory
Description	Information about a particular current UDP listener. INDEX { udpLocalAddress, udpLocalPort }
Sequence	::= { udpTable 1 } UdpEntry ::= SEQUENCE { udpLocalAddress IpAddress, udpLocalPort INTEGER (0..65535) }

udpLocalAddress

Syntax	IpAddress
Access	read-only
Status	mandatory
Description	The local IP address for this UDP listener. In the case of a UDP listener which is willing to accept datagrams for any IP interface associated with the node, the value 0.0.0.0 is used.
Sequence	::= { udpEntry 1 }

udpLocalPort

Syntax	INTEGER (0..65535)
Access	read-only
Status	mandatory
Description	The local port number for this UDP listener.
Sequence	::= { udpEntry 2 }

SNMP group

Implementation of the SNMP group is mandatory for all systems which support an SNMP protocol entity. Some of the objects defined below will be zero-valued in those SNMP implementations that are optimized to support only those functions specific to either a management agent or a management station. In particular, it should be observed that the objects below refer to an SNMP entity, and there may be several SNMP entities residing on a managed node (e.g., if the node is hosting acting as a management station).

snmpInPkts

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of Messages delivered to the SNMP entity from the transport service.
Sequence	::= { snmp 1 }

snmpOutPkts

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP Messages which were passed from the SNMP protocol entity to the transport service.
Sequence	::= { snmp 2 }

snmpInBadVersions

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP Messages which were delivered to the SNMP protocol entity and were for an unsupported SNMP version.

Sequence ::= { snmp 3 }

snmpInBadCommunityNames

Syntax Counter

Access read-only

Status mandatory

Description The total number of SNMP Messages delivered to the SNMP protocol entity which used a SNMP community name not known to said entity.

Sequence ::= { snmp 4 }

snmpInBadCommunityUses

Syntax Counter

Access read-only

Status mandatory

Description The total number of SNMP Messages delivered to the SNMP protocol entity which represented an SNMP operation which was not allowed by the SNMP community named in the Message.

Sequence ::= { snmp 5 }

snmpInASNParseErrs

Syntax Counter

Access read-only

Status mandatory

Description The total number of ASN.1 or BER errors encountered by the SNMP protocol entity when decoding received SNMP Messages.

Sequence ::= { snmp 6 }
- { snmp 7 } is not used

snmpInTooBigs

Syntax Counter

Access	read-only
Status	mandatory
Description	The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is `tooBig'.
Sequence	::= { snmp 8 }

snmpInNoSuchNames

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is `noSuchName'.
Sequence	::= { snmp 9 }

snmpInBadValues

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is `badValue'.
Sequence	::= { snmp 10 }

snmpInReadOnlys

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number valid SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is `readOnly'. It should be noted that it is a protocol error to generate an SNMP PDU which contains the value `readOnly' in the

error-status field, as such this object is provided as a means of detecting incorrect implementations of the SNMP.

Sequence ::= { snmp 11 }

snmpInGenErrs

Syntax Counter

Access read-only

Status mandatory

Description The total number of SNMP PDUs which were delivered to the SNMP protocol entity and for which the value of the error-status field is 'genErr'.

Sequence ::= { snmp 12 }

snmpInTotalReqVars

Syntax Counter

Access read-only

Status mandatory

Description The total number of MIB objects which have been retrieved successfully by the SNMP protocol entity as the result of receiving valid SNMP Get-Request and Get-Next PDUs.

Sequence ::= { snmp 13 }

snmpInTotalSetVars

Syntax Counter

Access read-only

Status mandatory

Description The total number of MIB objects which have been altered successfully by the SNMP protocol entity as the result of receiving valid SNMP Set-Request PDUs.

Sequence ::= { snmp 14 }

snmpInGetRequests

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP Get-Request PDUs which have been accepted and processed by the SNM protocol entity.
Sequence	::= { snmp 15 }

snmpInGetNexts

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP Get-Next PDUs which have been accepted and processed by the SNMP protocol entity.
Sequence	::= { snmp 16 }

snmpInSetRequests

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP Set-Request PDUs which have been accepted and processed by the SNMP protocol entity.
Sequence	::= { snmp 17 }

snmpInGetResponses

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP Get-Response PDUs which have been accepted and processed by the SNMP protocol entity.

Sequence ::= { snmp 18 }

snmpInTraps

Syntax Counter

Access read-only

Status mandatory

Description The total number of SNMP Trap PDUs which have been accepted and processed by the SNMP protocol entity.

Sequence ::= { snmp 19 }

snmpOutTooBigs

Syntax Counter

Access read-only

Status mandatory

Description The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is 'tooBig.'

Sequence ::= { snmp 20 }

snmpOutNoSuchNames

Syntax Counter

Access read-only

Status mandatory

Description The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status is 'noSuchName'.

Sequence ::= { snmp 21 }

snmpOutBadValues

Syntax Counter

Access read-only

Status	mandatory
Description	The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is 'badValue'.
Sequence	::= { snmp 22 } { snmp 23 } is not used

snmpOutGenErrs

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP PDUs which were generated by the SNMP protocol entity and for which the value of the error-status field is 'genErr'.
Sequence	::= { snmp 24 }

snmpOutGetRequests

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP Get-Request PDUs which have been generated by the SNMP protocol entity.
Sequence	::= { snmp 25 }

snmpOutGetNexts

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP Get-Next PDUs which have been generated by the SNMP protocol entity.
Sequence	::= { snmp 26 }

snmpOutSetRequests

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP Set-Request PDUs which have been generated by the SNMP protocol entity.
Sequence	::= { snmp 27 }

snmpOutGetResponses

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP Get-Response PDUs which have been generated by the SNMP protocol entity.
Sequence	::= { snmp 28 }

snmpOutTraps

Syntax	Counter
Access	read-only
Status	mandatory
Description	The total number of SNMP Trap PDUs which have been generated by the SNMP protocol entity.
Sequence	::= { snmp 29 }

snmpEnableAuthenTraps

Syntax	INTEGER { enabled(1), disabled(2) }
Access	read-write
Status	mandatory
Description	Indicates whether the SNMP agent process is permitted to generate authentication-failure traps. The value of this object overrides any

configuration information; as such, it provides a means whereby all authentication-failure traps may be disabled.

Note that it is strongly recommended that this object be stored in non-volatile memory so that it remains constant between re-initializations of the network management system.

Sequence ::= { snmp 30 }
END

FCFE.MIB

April 24, 2000

Fabric Element Management MIB, Version 1.10 (same as version 1.9), as per Internet Draft, "Definitions of Managed Objects for the Fabric Element in Fibre Channel Standard", <draft-teow-fabric-element-mib-03.txt>, July 10, 1998.

This is edited for McDATA implementation.

(1) The following variables have been changed from read-write to read-only:

fcFabricName, fcElementName, fcFeModuleName (Config group), fcFxPortAdminMode, fcFxPortPhysRttov, fcFxPortBbCreditModel (Operational Group).

(2) The following deprecated objects are not supported:

FPortFlogiTable { fcFeOp 2 }

(3) The following textual conventions have been changed slightly:

MilliSecond and MicroSecond such that the maximum value is 2147383647 ($2^{31} - 1$) instead of 4294967295 ($2^{32} - 1$) due to the restriction of the MIB compiler (mibcomp). Brocade plans to propose these minor changes to the next version of the Internet Draft.

This MIB module is equivalent to femib.smiv2 but it's in SMIV1.

```

FCFABRIC-ELEMENT-MIB DEFINITIONS ::= BEGIN

IMPORTS
    experimental, Counter, Gauge, TimeTicks
        FROM RFC1155-SMI;

fibreChannel OBJECT IDENTIFIER ::= { experimental 42 }
fcFabric MODULE-IDENTITY
    ::= { fibreChannel 2 }
fcFabric OBJECT IDENTIFIER ::= { fibreChannel 2 }
Fabric Element
fcFe OBJECT IDENTIFIER ::= { fcFabric 1 }

```

Groups under fcFe

fcFeConfig	OBJECT IDENTIFIER	::= { fcFe 1 }
fcFeOp	OBJECT IDENTIFIER	::= { fcFe 2 }
fcFeError	OBJECT IDENTIFIER	::= { fcFe 3 }
fcFeAcct	OBJECT IDENTIFIER	::= { fcFe 4 }
fcFeCap	OBJECT IDENTIFIER	::= { fcFe 5 }

Type definitions.

DisplayString	::= OCTET STRING
Milliseconds	::= INTEGER (0..2147383647) -- $2^{31} - 1$
MicroSeconds	::= INTEGER (0..2147383647)
FcNameId	::= OCTET STRING (SIZE (8))

Worldwide Name or Fibre Channel Name associated with an FC entity. It's a Network_Destination_ID or Network_Source_ID composed of a value up to 60 bits wide, occupying the remaining 8 bytes while the first nibble identifies the format of the Name_Identifier with hex values:

- 0: ignored
- 1: IEEE 48-bit address,
- 2: IEEE extended,
- 3: Locally assigned,

4: 32-bit IP address

FabricName ::= FcNameId

The Name Identifier of a Fabric. Each Fabric shall provide a unique Fabric Name. Only the following formats are allowed: IEEE48, and Local.

FcPortName ::= FcNameId

The Name Identifier associated with a port Only the following formats are allowed: IEEE48, IEEE extended, and Local.

FcAddressId ::= OCTET STRING (SIZE (3))

Fibre Channel Address Identifier. A 24-bit value unique within the address space of a Fabric

FcRxDataFieldSize ::= INTEGER (128..2112)

Receive Data_Field Size

FcBbCredit ::= INTEGER (0..32767)

Buffer-to-buffer Credit

FC-PH version

FcphVersion ::= INTEGER (0..255)

Class 1 Stacked Connect Support/Mode

FcStackedConnMode ::= INTEGER {

none (1),
transparent (2),
lockedDown (3) }

Class of Service Capability Set

FcCosCap ::= INTEGER (0..127)

bit 0 Class F

bit 1 Class 1

bit 2 Class 2

bit 3 Class 3

bit 4 Class 4

bit 5 Class 5

bit 6 Class 6

bit 7 reserved for future

FC-0 Baud Rates

```
Fc0BaudRate ::= INTEGER {
    other (1), -- none of below
    oneEighth (2), -- 155 Mbaud (12.5MB/s)
    quarter (4), -- 266 Mbaud (25.0MB/s)
    half (8), -- 532 Mbaud (50.0MB/s)
    full (16), -- 1 Gbaud (100MB/s)
    double (32), -- 2 Gbaud (200MB/s)
    quadruple (64) -- 4 Gbaud (400MB/s) }
```

Baud Rate Capability Set

```
Fc0BaudRateCap ::= INTEGER (0..127)

bit 0 other
bit 1 oneEighth
bit 2 quarter
bit 3 half
bit 4 full
bit 5 double
bit 6 quadruple
bit 7 reserved for future
```

FC-0 Media Capability Set

```
Fc0MediaCap ::= INTEGER (0..65535)

bit 0 unknown
bit 1 single mode fibre (sm)
bit 2 multi-mode fibre 50 micron (m5)
bit 3 multi-mode fibre 62.5 micron (m6)
bit 4 video cable (tv)
bit 5 miniature cable (mi)
```

bit 6 shielded twisted pair (stp)

bit 7 twisted wire (tw)

bit 8 long video (lv)

bits 9-15 reserved for future use

A specific FC-0 medium type associated with a port

```
Fc0Medium ::= INTEGER {  
    unknown (1),  
    sm (2),  
    m5 (4),  
    m6 (8),  
    tv (16),  
    mi (32),  
    stp (64),  
    tw (128),  
    lv (256) }
```

The FC-0 transmitter type of a port

```
Fc0TxType ::= INTEGER {  
    unknown (1),  
    longWaveLaser (2),    (LL)  
    shortWaveLaser (3),    (SL)  
    longWaveLED (4),    (LE)  
    electrical (5), (EL)  
    shortWaveLaser-noOFC (6)(SN) }
```

The FC-0 distance range associated with a port transmitter

```
Fc0Distance ::= INTEGER { unknown (1), long (2), intermediate (3),  
    short (4) }
```

Module and Port Capacity

FcFeModuleCapacity ::= INTEGER (1..256)

FcFeFxPortCapacity ::= INTEGER (1..256)

Module, FxPort and NxPort Index

FcFeModuleIndex ::= INTEGER (1..256)

FcFeFxPortIndex ::= INTEGER (1..256)

FcFeNxPortIndex ::= INTEGER (1..126)

Port Mode

FcFxPortMode ::= INTEGER { unknown (1), fPort (2), flPort (3) }

BB_Credit Model

FcBbCreditModel ::= INTEGER { regular (1), alternate (2) }

Configuration group

This group consists of scalar objects and tables. It contains the configuration and service parameters of the Fabric Element and the FxPorts. The group represents a set of parameters associated with the Fabric Element or an FxPort to support its NxPorts. Implementation of this group is mandatory.

fcFabricName

Syntax	FabricName
Access	read-only -- instead of read-write
Status	mandatory
Description	The Name_Identifier of the Fabric to which this Fabric Element belongs.
Sequence	::= { fcFeConfig 1 }

fcElementName

Syntax	FcNameId
Access	read-only -- instead of read-write

Status	mandatory
Description	The Name_Identifier of the Fabric Element.
Sequence	::= { fcFeConfig 2 }

fcFeModuleCapacity

Syntax	FcFeModuleCapacity
Access	read-only
Status	mandatory
Description	The maximum number of modules in the Fabric Element, regardless of their current state.
Sequence	::= { fcFeConfig 3 }

The Module Table. This table contains one entry for each module, information of the modules.

fcFeModuleTable

Syntax	SEQUENCE OF FcFeModuleEntry
Access	not-accessible
Status	mandatory
Description	A table that contains, one entry for each module in the Fabric Element, information of the modules.
Sequence	::= { fcFeConfig 4 }

fcFeModuleEntry

Syntax	FcFeModuleEntry
Access	not-accessible
Status	mandatory
Description	An entry containing the configuration parameters of a module. INDEX { fcFeModuleIndex }
Sequence	::= { fcFeModuleTable 1 } FcFeModuleEntry ::=

```
SEQUENCE {  
    fcFeModuleIndex          FcFeModuleIndex,  
    fcFeModuleDescr          DisplayString,  
    fcFeModuleObjectID       OBJECT IDENTIFIER,  
    fcFeModuleOperStatus     INTEGER,  
    fcFeModuleLastChange     TimeTicks,  
    fcFeModuleFxpPortCapacity FcFeFxpPortCapacity,  
    fcFeModuleName           FcNameId }
```

fcFeModuleIndex

Syntax	FcFeModuleIndex
Access	read-only
Status	mandatory
Description	This object identifies the module within the Fabric Element for which this entry contains information. This value is never greater than fcFeModuleCapacity.
Sequence	::= { fcFeModuleEntry 1 }

fcFeModuleDescr

Syntax	DisplayString (SIZE(256))
Access	read-only
Status	mandatory
Description	A textual description of the module. This value should include the full name and version identification of the module. It should contain printable ASCII characters.
Sequence	::= { fcFeModuleEntry 2 }

fcFeModuleObjectID

Syntax	OBJECT IDENTIFIER
Access	read-only
Status	mandatory

Description	<p>The vendor's authoritative identification of the module. This value may be allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides a straight-forward and unambiguous means for determining what kind of module is being managed.</p> <p>For example, this object could take the value 1.3.6.1.4.1.99649.3.9 if vendor 'Neufe Inc.' was assigned the subtree 1.3.6.1.4.1.99649, and had assigned the identifier 1.3.6.1.4.1.99649.3.9 to its 'FeFiFo-16 PlugInCard.'</p>
Sequence	::= { fcFeModuleEntry 3 }

fcFeModuleOperStatus

Syntax	INTEGER { online (1), -- functional offline (2), -- not available testing (3), -- under testing faulty (4) -- defective }
Access	read-only
Status	mandatory
Description	<p>This object indicates the operational status of the module:</p> <p>online(1) the module is functioning properly; offline(2) the module is not available; testing(3) the module is under testing; and faulty(4) the module is defective in some way."</p>
Sequence	::= { fcFeModuleEntry 4 }

fcFeModuleLastChange

Syntax	TimeTicks
Access	read-only
Status	mandatory
Description	<p>This object contains the value of sysUpTime when the module entered its current operational status. A value of zero indicates that the operational status of the module has not changed since the agent last restarted.</p>

Sequence ::= { fcFeModuleEntry 5 }

fcFeModuleFxPortCapacity

Syntax FcFeFxPortCapacity

Access read-only

Status mandatory

Description The number of FxPort that can be contained within the module. Within each module, the ports are uniquely numbered in the range from 1 to fcFeModuleFxPortCapacity inclusive. However, the numbers are not required to be contiguous.

Sequence ::= { fcFeModuleEntry 6 }

fcFeModuleName

Syntax FcNameId

Access read-only -- instead of read-write

Status mandatory

Description The Name_Identifier of the module.

Sequence ::= { fcFeModuleEntry 7 }

FxPort Configuration Table This table contains, one entry for each FxPort, configuration parameters of the ports.

fcFxConfTable

Syntax SEQUENCE OF FcFxConfEntry

Access not-accessible

Status mandatory

Description A table that contains, one entry for each FxPort in the Fabric Element, configuration and service parameters of the FxPorts."

Sequence ::= { fcFeConfig 5 }

fcFxConfEntry

Syntax	FcFxConfEntry
Access	not-accessible
Status	mandatory
Description	An entry containing the configuration and service parameters of a FxPort. INDEX { fcFxConfModuleIndex, fcFxConfFxPortIndex }
Sequence	::= { fcFxConfTable 1 } FcFxConfEntry ::= SEQUENCE { fcFxConfModuleIndex FcFeModuleIndex, fcFxConfFxPortIndex , FcFeFxPortIndex fcFxPortName FcPortName,

FxPort common service parameters

fcFxPortFcphVersionHigh	FcphVersion,
fcFxPortFcphVersionLow	FcphVersion,
fcFxPortBbCredit	FcBbCredit,
fcFxPortRxBufSize	FcRxDataFieldSize,
fcFxPortRatov	Milliseconds,
fcFxPortEdtov	Milliseconds,

FxPort class service parameters

fcFxPortCosSupported	FcCosCap,
fcFxPortIntermixSupported	INTEGER,
fcFxPortStackedConnMode	FcStackedConnMode,
fcFxPortClass2SeqDeliv	INTEGER,
fcFxPortClass3SeqDeliv	INTEGER,

Other configuration parameters

fcFxCPortHoldTime	MicroSeconds,
fcFxCPortBaudRate	Fc0BaudRate,
fcFxCPortMedium	Fc0Medium,
fcFxCPortTxType	Fc0TxType,
fcFxCPortDistance	Fc0Distance }

fcFxCConfModuleIndex

Syntax	FcFeModuleIndex
Access	read-only
Status	mandatory
Description	This object identifies the module containing the FxCPort for which this entry contains information.
Sequence	::= { fcFxCConfEntry 1 }

fcFxCConfFxCPortIndex

Syntax	FcFeFxCPortIndex
Access	read-only
Status	mandatory
Description	This object identifies the FxCPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxCPort until the module is re-initialized.
Sequence	::= { fcFxCConfEntry 2 }

fcFxCPortName

Syntax	FcPortName
Access	read-only
Status	mandatory
Description	The name identifier of this FxCPort. Each FxCPort has a unique port name within the address space of the Fabric.

Sequence ::= { fcFxConfEntry 3 }

FxPort common service parameters

fcFxPortFcphVersionHigh

Syntax	FcphVersion
Access	read-only
Status	mandatory
Description	The highest or most recent version of FC-PH that the FxPort is configured to support.
Sequence	::= { fcFxConfEntry 4 }

fcFxPortFcphVersionLow

Syntax	FcphVersion
Access	read-only
Status	mandatory
Description	The lowest or earliest version of FC-PH that the FxPort is configured to support.
Sequence	::= { fcFxConfEntry 5 }

fcFxPortBbCredit

Syntax	FcBbCredit
Access	read-only
Status	mandatory
Description	The total number of receive buffers available for holding Class 1 connect-request, Class 2 or 3 frames from the attached NxPort. It is for buffer-to-buffer flow control in the direction from the attached NxPort (if applicable) to FxPort.
Sequence	::= { fcFxConfEntry 6 }

fcFxpPortRxBufSize

Syntax	FcRxDataFieldSize
Access	read-only
Status	mandatory
Description	The largest Data_Field Size (in octets) for an FT_1 frame that can be received by the FxPort.
Sequence	::= { fcFxpConfEntry 7 }

fcFxpPortRatov

Syntax	Milliseconds
Access	read-only
Status	mandatory
Description	The Resource_Allocation_Timeout Value configured for the FxPort. This is used as the timeout value for determining when to reuse an NxPort resource such as a Recovery_Qualifier. It represents E_D_TOV (see next object) plus twice the maximum time that a frame may be delayed within the Fabric and still be delivered.
Sequence	::= { fcFxpConfEntry 8 }

fcFxpPortEdtov

Syntax	Milliseconds
Access	read-only
Status	mandatory
Description	The E_D_TOV value configured for the FxPort. The Error_Detect_Timeout Value is used as the timeout value for detecting an error condition.
Sequence	::= { fcFxpConfEntry 9 }

FxPort class service parameters

fcFxPortCosSupported

Syntax	FcCosCap
Access	read-only
Status	mandatory
Description	A value indicating the set of Classes of Service supported by the FxPort.
Sequence	::= { fcFxConfEntry 10 }

fcFxPortIntermixSupported

Syntax	INTEGER { yes (1), no (2) }
Access	read-only
Status	mandatory
Description	A flag indicating whether or not the FxPort supports an Intermixed Dedicated Connection.
Sequence	::= { fcFxConfEntry 11 }

fcFxPortStackedConnMode

Syntax	FcStackedConnMode
Access	read-only
Status	mandatory
Description	A value indicating the mode of Stacked Connect supported by the FxPort.
Sequence	::= { fcFxConfEntry 12 }

fcFxPortClass2SeqDeliv

Syntax	INTEGER { yes (1), no (2) }
Access	read-only
Status	mandatory

Description A flag indicating whether or not Class 2 Sequential Delivery is supported by the FxPort.

Sequence ::= { fcFxConfEntry 13 }

fcFxPortClass3SeqDeliv

Syntax INTEGER { yes (1), no (2) }

Access read-only

Status mandatory

Description A flag indicating whether or not Class 3 Sequential Delivery is supported by the FxPort.

Sequence ::= { fcFxConfEntry 14 }

Other FxPort parameters

fcFxPortHoldTime

Syntax MicroSeconds

Access read-only

Status mandatory

Description The maximum time (in microseconds) that the FxPort shall hold a frame before discarding the frame if it is unable to deliver the frame. The value 0 means that the FxPort does not support this parameter."

Sequence ::= { fcFxConfEntry 15 }

fcFxPortBaudRate

Syntax Fc0BaudRate

Access read-only

Status deprecated

Description The FC-0 baud rate of the FxPort.

Sequence ::= { fcFxConfEntry 16 }

fcFxPortMedium

Syntax	Fc0Medium
Access	read-only
Status	deprecated
Description	The FC-0 medium of the FxPort.
Sequence	::= { fcFxConfEntry 17 }

fcFxPortTxType

Syntax	Fc0TxType
Access	read-only
Status	deprecated
Description	The FC-0 transmitter type of the FxPort.
Sequence	::= { fcFxConfEntry 18 }

fcFxPortDistance

Syntax	Fc0Distance
Access	read-only
Status	deprecated
Description	The FC-0 distance range of the FxPort transmitter.
Sequence	::= { fcFxConfEntry 19 }

Operation group

This group consists of tables that contains operational status and established service parameters for the Fabric Element and the attached NxPorts. Implementation of this group is mandatory.

The FxPort Operation table

This table contains, one entry for each FxPort, the operational status and parameters of the FxPorts.

fcFxPortOperTable

Syntax	SEQUENCE OF FcFxPortOperEntry
Access	not-accessible
Status	mandatory
Description	A table that contains, one entry for each FxPort in the Fabric Element, operational status and parameters of the FxPorts.
Sequence	::= { fcFeOp 1 }

fcFxPortOperEntry

Syntax	FcFxPortOperEntry
Access	not-accessible
Status	mandatory
Description	An entry containing operational status and parameters of a FxPort." INDEX { fcFxPortOperModuleIndex, fcFxPortOperFxPortIndex }
Sequence	::= { fcFxPortOperTable 1 } FcFxPortOperEntry ::= SEQUENCE { fSyntax cFxPortOperModuleIndex FcFeModuleIndex, fcFxPortOperFxPortIndex FcFeFxPortIndex, fcFxPortID FcAddressId, fcFPortAttachedPortName FcPortName, fcFPortConnectedPort FcAddressId,

fcFxPortBbCreditAvailable	Gauge,
fcFxPortOperMode	FcFxPortMode,
fcFxPortAdminMode	FcFxPortMode }

fcFxPortOperModuleIndex

Syntax	FcFeModuleIndex
Access	read-only
Status	mandatory
Description	This object identifies the module containing the FxPort for which this entry contains information.
Sequence	::= { fcFxPortOperEntry 1 }

fcFxPortOperFxPortIndex

Syntax	FcFeFxPortIndex
Access	read-only
Status	mandatory
Description	This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized."
Sequence	::= { fcFxPortOperEntry 2 }

fcFxPortID

Syntax	FcAddressId
Access	read-only
Status	mandatory
Description	The address identifier by which this FxPort is identified within the Fabric. The FxPort may assign its address identifier to its attached NxPort(s) during Fabric Login."
Sequence	::= { fcFxPortOperEntry 3 }

fcFPortAttachedPortName

Syntax	FcPortName
Access	read-only
Status	deprecated
Description	The port name of the attached N_Port, if applicable. If the value of this object is '0000000000000000'H, this FxPort has no NxPort attached to it. This variable has been deprecated and may be implemented for backward compability.
Sequence	::= { fcFxPortOperEntry 4 }

fcFPortConnectedPort

Syntax	FcAddressId
Access	read-only
Status	deprecated
Description	The address identifier of the destination FxPort with which this FxPort is currently engaged in a either a Class 1 or loop connection. If the value of this object is '000000'H, this FxPort is not engaged in a connection. This variable has been deprecated and may be implemented for backward compability.
Sequence	::= { fcFxPortOperEntry 5 }

fcFxPortBbCreditAvailable

Syntax	Gauge
Access	read-only
Status	mandatory
Description	The number of buffers currently available for receiving frames from the attached port in the buffer-to-buffer flow control. The value should be less than or equal to fcFxPortBbCredit.
Sequence	::= { fcFxPortOperEntry 6 }

fcFxpPortOperMode

Syntax	FcFxpPortMode
Access	read-only
Status	mandatory
Description	The current operational mode of the FxPort.
Sequence	::= { fcFxpPortOperEntry 7 }

fcFxpPortAdminMode

Syntax	FcFxpPortMode
Access	read-only -- instead of read-write
Status	mandatory
Description	The desired operational mode of the FxPort.
Sequence	::= { fcFxpPortOperEntry 8 }

F_Port Fabric Login table

This table contains, one entry for each F_Port in the Fabric Element, the Service Parameters that have been established from the most recent Fabric Login (implicit or explicit).

NOTE WELL:

This table is deprecated since FEMIB v1.9. It is not supported in Silkorm agent after firmware v1.5. Instead, the new table, FxPort Fabric Login Table (to follow after FxPort Physical Level Table), is supported.

FxPort Physical Level table

This table contains one entry for each FxPort in the Fabric Element, the physical level status and parameters of the FxPorts.

fcFxpPortPhysTable

Syntax	SEQUENCE OF FcFxpPortPhysEntry
Access	not-accessible
Status	mandatory

Description A table that contains, one entry for each FxPort in the Fabric Element, physical level status and parameters of the FxPorts."

Sequence ::= { fcFeOp 3 }

fcFxPortPhysEntry

Syntax FcFxPortPhysEntry

Access not-accessible

Status mandatory

Description An entry containing physical level status and parameters of a FxPort.
INDEX { fcFxPortPhysModuleIndex, fcFxPortPhysFxPortIndex }

Sequence ::= { fcFxPortPhysTable 1 }

FcFxPortPhysEntry ::=

SEQUENCE {

fcFxPortPhysModuleIndex FcFeModuleIndex,

fcFxPortPhysFxPortIndex FcFeFxPortIndex,

fcFxPortPhysAdminStatus INTEGER,

fcFxPortPhysOperStatus INTEGER,

fcFxPortPhysLastChange TimeTicks,

fcFxPortPhysRttov MilliSeconds }

fcFxPortPhysModuleIndex

Syntax FcFeModuleIndex

Access read-only

Status mandatory

Description This object identifies the module containing the FxPort for which this entry contains information.

Sequence ::= { fcFxPortPhysEntry 1 }

fcFxPortPhysFxPortIndex

Syntax FcFeFxPortIndex

Access read-only

Status	mandatory
Description	This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.
Sequence	::= { fcFxPortPhysEntry 2 }

fcFxPortPhysAdminStatus

Syntax	INTEGER { online (1), -- place port online offline (2), -- take port offline testing (3) -- initiate test procedures }
Access	read-write
Status	mandatory
Description	<p>The desired state of the FxPort. A management station may place the FxPort in a desired state by setting this object accordingly. The testing(3) state indicates that no operational frames can be passed. When a Fabric Element initializes, all FxPorts start with fcFxPortPhysAdminStatus in the offline(2) state.</p> <p>As the result of either explicit management action or per configuration information accessible by the Fabric Element, fcFxPortPhysAdminStatus is then changed to either the online(1) or testing(3) states, or remains in the offline state.</p>
Sequence	::= { fcFxPortPhysEntry 3 }

fcFxPortPhysOperStatus

Syntax	INTEGER { online (1), -- Login may proceed offline (2), -- Login cannot proceed testing (3), -- port is under test link-failure (4) -- failure after online/testing Other values may be used to indicate diagnostic for failed test. }
Access	read-only

Status	mandatory
Description	The current operational status of the FxPort. The testing(3) indicates that no operational frames can be passed. If fcFxPortPhysAdminStatus is offline(2) then fcFxPortPhysOperStatus should be offline(2). If fcFxPortPhysAdminStatus is changed to online(1) then fcFxPortPhysOperStatus should change to online(1) if the FxPort is ready to accept Fabric Login request from the attached NxPort; it should proceed and remain in the link-failure(4) state if and only if there is a fault that prevents it from going to the online(1) state.
Sequence	::= { fcFxPortPhysEntry 4 }

fcFxPortPhysLastChange

Syntax	TimeTicks
Access	read-only
Status	mandatory
Description	The value of sysUpTime at the time the FxPort entered its current operational status. A value of zero indicates that the FxPort's operational status has not changed since the agent last restarted.
Sequence	::= { fcFxPortPhysEntry 5 }

fcFxPortPhysRttov

Syntax	Milliseconds
Access	read-only -- instead of read-write
Status	mandatory
Description	The Receiver_Transmitter_Timeout value of the FxPort. This is used by the receiver logic to detect Loss of Synchronization.
Sequence	::= { fcFxPortPhysEntry 6 }

FxPort Fabric Login table

This table contains, one entry for each FxPort in the Fabric Element, the Service Parameters that have been established from the most recent Fabric Login, implicit or explicit.

fcFxlogiTable

Syntax	SEQUENCE OF FcFxlogiEntry
Access	not-accessible
Status	mandatory
Description	A table that contains, one entry for each FxPort in the Fabric Element, services parameters established from the most recent Fabric Login, explicit or implicit.
Sequence	::= { fcFeOp 4 }

fcFxlogiEntry

Syntax	FcFxlogiEntry
Access	not-accessible
Status	mandatory
Description	An entry containing service parameters established from a successful Fabric Login."
	INDEX { fcFxlogiModuleIndex, fcFxlogiFxPortIndex, fcFxlogiNxPortIndex }
Sequence	::= { fcFxlogiTable 1 }
	FcFxlogiEntry ::=
	SEQUENCE {
	fcFxlogiModuleIndex FcFeModuleIndex,
	fcFxlogiFxPortIndex FcFeFxPortIndex,
	fcFxlogiNxPortIndex FcFeNxPortIndex,
	fcFxPortFcphVersionAgreed FcphVersion,
	fcFxPortNxPortBbCredit FcBbCredit,
	fcFxPortNxPortRxDataFieldSize FcRxDataFieldSize,

fcFxpPortCosSuppAgreed	FcCosCap,
fcFxpPortIntermixSuppAgreed	INTEGER,
fcFxpPortStackedConnModeAgreed	FcStackedConnMode,
fcFxpPortClass2SeqDelivAgreed	INTEGER,
fcFxpPortClass3SeqDelivAgreed	INTEGER,
fcFxpPortNxPortName	FcPortName,
fcFxpPortConnectedNxPort	FcAddressId,
fcFxpPortBbCreditModel	FcBbCreditModel }

fcFxplogiModuleIndex

Syntax	FcFeModuleIndex
Access	read-only
Status	mandatory
Description	This object identifies the module containing the FxpPort for which this entry contains information.
Sequence	::= { fcFxplogiEntry 1 }

fcFxplogiFxpPortIndex

Syntax	FcFeFxpPortIndex
Access	read-only
Status	mandatory
Description	This object identifies the FxpPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxpPort until the module is re-initialized.
Sequence	::= { fcFxplogiEntry 2 }

fcFxplogiNxPortIndex

Syntax	FcFeNxPortIndex
Access	read-only
Status	mandatory

Description The object identifies the associated NxPort in the attachment for which the entry contains information.

Sequence ::= { fcFxlogiEntry 3 }

fcFxPortFcphVersionAgreed

Syntax FcphVersion

Access read-only

Status mandatory

Description The version of FC-PH that the FxPort has agreed to support from the Fabric Login

Sequence ::= { fcFxlogiEntry 4 }

fcFxPortNxPortBbCredit

Syntax FcBbCredit

Access read-only

Status mandatory

Description The total number of buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames to be transmitted to the attached NxPort. It is for buffer- to-buffer flow control in the direction from FxPort to NxPort. The buffer-to-buffer flow control mechanism is indicated in the respective fcFxPortBbCreditModel. [1](23.6.2.2)

Sequence ::= { fcFxlogiEntry 5 }

fcFxPortNxPortRxDataFieldSize

Syntax FcRxDataFieldSize

Access read-only

Status mandatory

Description The Receive Data Field Size of the attached NxPort. This is a binary value that specifies the largest Data Field Size for an FT_1 frame that can be received by the NxPort. The value is in number of bytes and ranges from 128 to 2112 inclusive.

Sequence ::= { fcFxlogiEntry 6 }

fcFxPortCosSuppAgreed

Syntax	FcCosCap
Access	read-only
Status	mandatory
Description	A variable indicating that the attached NxPort has requested the FxPort for the support of classes of services and the FxPort has granted the request.
Sequence	::= { fcFxlogiEntry 7 }

fcFxPortIntermixSuppAgreed

Syntax	INTEGER { yes (1), no (2) }
Access	read-only
Status	mandatory
Description	A variable indicating that the attached NxPort has requested the FxPort for the support of Intermix and the FxPort has granted the request. This flag is only valid if Class 1 service is supported.
Sequence	::= { fcFxlogiEntry 8 }

fcFxPortStackedConnModeAgreed

Syntax	FcStackedConnMode
Access	read-only
Status	mandatory
Description	A variable indicating whether the FxPort has agreed to support stacked connect from the Fabric Login. This is only meaningful if Class 1 service has been agreed.
Sequence	::= { fcFxlogiEntry 9 }

fcFxPortClass2SeqDelivAgreed

Syntax	INTEGER { yes (1), no (2) }
Access	read-only
Status	mandatory

Description A variable indicating whether the FxPort has agreed to support Class 2 sequential delivery from the Fabric Login. This is only meaningful if Class 2 service has been agreed.

Sequence ::= { fcFxlogiEntry 10 }

fcFxPortClass3SeqDelivAgreed

Syntax INTEGER { yes (1), no (2) }

Access read-only

Status mandatory

Description A flag indicating whether the FxPort has agreed to support Class 3 sequential delivery from the Fabric Login. This is only meaningful if Class 3 service has been agreed.

Sequence ::= { fcFxlogiEntry 11 }

fcFxPortNxPortName

Syntax FcPortName

Access read-only

Status mandatory

Description The port name of the attached NxPort, if applicable. If the value of this object is '0000000000000000'H, this FxPort has no NxPort attached to it.

Sequence ::= { fcFxlogiEntry 12 }

fcFxPortConnectedNxPort

Syntax FcAddressId

Access read-only

Status mandatory

Description The address identifier of the destination FxPort with which this FxPort is currently engaged in a either a Class 1 or loop connection. If the value of this object is '000000'H, this FxPort is not engaged in a connection.

Sequence ::= { fcFxlogiEntry 13 }

fcFxpPortBbCreditModel

Syntax	FcBbCreditModel
Access	read-only -- instead of read-write
Status	mandatory
Description	This object identifies the BB_Credit model used by the FxPort. The regular model refers to the Buffer-to-Buffer flow control mechanism defined in FC-PH [1] is used between the F_Port and the N_Port. For FL_Ports, the Alternate Buffer-to-Buffer flow control mechanism as defined in FC-AL [4] is used between the FL_Port and any attached NL_Ports.
Sequence	::= { fcFxplogiEntry 14 }

Error group

This group consists of tables that contain information about the various types of errors detected. The management station may use the information in this group to determine the quality of the link between the FxPort and its attached NxPort. Implementation of this group is optional.

FxPort Error table

This table contains, one entry for each FxPort in the Fabric Element, counters recording numbers of errors detected since the management agent re-initialized. The first 6 columnar objects after the port index corresponds to the counters in the Link Error Status Block ([1](29.8)).

fcFxpPortErrorTable

Syntax	SEQUENCE OF FcFxpPortErrorEntry
Access	not-accessible
Status	mandatory
Description	A table that contains, one entry for each FxPort, counters that record the numbers of errors detected.
Sequence	::= { fcFeError 1 }

fcFxPortErrorEntry

Syntax	FcFxPortErrorEntry
Access	not-accessible
Status	mandatory
Description	An entry containing error counters of a FxPort. INDEX { fcFxPortErrorModuleIndex, fcFxPortErrorFxPortIndex }
Sequence	::= { fcFxPortErrorTable 1 } FcFxPortErrorEntry ::= SEQUENCE { fcFxPortErrorModuleIndex FcFeModuleIndex, fcFxPortErrorFxPortIndex FcFeFxPortIndex, fcFxPortLinkFailures Counter, fcFxPortSyncLosses Counter, fcFxPortSigLosses Counter, fcFxPortPrimSeqProtoErrors Counter, fcFxPortInvalidTxWords Counter, fcFxPortInvalidCrcs Counter, fcFxPortDelimiterErrors Counter, fcFxPortAddressIdErrors Counter, fcFxPortLinkResetIns Counter, fcFxPortLinkResetOuts ounter, fcFxPortOlsIns Counter, fcFxPortOlsOuts Counter }

fcFxPortErrorModuleIndex

Syntax	FcFeModuleIndex
Access	read-only
Status	mandatory
Description	This object identifies the module containing the FxPort for which this entry contains information.

Sequence ::= { fcFxpPortErrorEntry 1 }

fcFxpPortErrorFxpPortIndex

Syntax FcFeFxpPortIndex

Access read-only

Status mandatory

Description This object identifies the FxpPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxpPort until the module is re-initialized.

Sequence ::= { fcFxpPortErrorEntry 2 }

fcFxpPortLinkFailures

SYNTAX Counter

Access read-only

Status mandatory

Description The number of link failures detected by this FxpPort.

Sequence ::= { fcFxpPortErrorEntry 3 }

fcFxpPortSyncLosses

SYNTAX Counter

Access read-only

Status mandatory

Description The number of loss of synchronization detected by the FxpPort.

Sequence ::= { fcFxpPortErrorEntry 4 }

fcFxpPortSigLosses

SYNTAX Counter

Access read-only

Status mandatory

Description The number of loss of signal detected by the FxpPort.

Sequence ::= { fcFxpPortErrorEntry 5 }

fcFxpPortPrimSeqProtoErrors

SYNTAX Counter
 Access read-only
 Status mandatory
 Description The number of primitive sequence protocol errors detected by the FxPort.
 Sequence ::= { fcFxpPortErrorEntry 6 }

fcFxpPortInvalidTxWords

SYNTAX Counter
 Access read-only
 Status mandatory
 Description The number of invalid transmission word detected by the FxPort.
 Sequence ::= { fcFxpPortErrorEntry 7 }

fcFxpPortInvalidCrcs

SYNTAX Counter
 Access read-only
 Status mandatory
 Description The number of invalid CRC detected by this FxPort.
 Sequence ::= { fcFxpPortErrorEntry 8 }

fcFxpPortDelimiterErrors

SYNTAX Counter
 Access read-only
 Status mandatory
 Description The number of Delimiter Errors detected by this FxPort.
 Sequence ::= { fcFxpPortErrorEntry 9 }

fcFxpPortAddressIdErrors

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of address identifier errors detected by this FxPort.
Sequence	::= { fcFxpPortErrorEntry 10 }

fcFxpPortLinkResetIns

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Link Reset Protocol received by this FxPort from the attached NxPort.
Sequence	::= { fcFxpPortErrorEntry 11 }

fcFxpPortLinkResetOuts

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Link Reset Protocol issued by this FxPort to the attached NxPort.
Sequence	::= { fcFxpPortErrorEntry 12 }

fcFxpPortOlsIns

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Offline Sequence received by this FxPort.
Sequence	::= { fcFxpPortErrorEntry 13 }

fcFxpPortOlsOuts

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Offline Sequence issued by this FxPort.
Sequence	::= { fcFxpPortErrorEntry 14 }

Accounting Groups

- (1) Class 1 Accounting Group,
- (2) Class 2 Accounting Group, and
- (3) Class 3 Accounting Group.

Each group consists of a table that contains accounting information for the FxPorts in the Fabric Element. Implementation of each group is optional.

Class 1 Accounting table

This table contains, one entry for each FxPort in the Fabric Element, Counters for certain types of events occurred in the the FxPorts since the the management agent has re-initialized.

Implementation of this group is optional.

fcFxpPortC1AcctTable

Syntax	SEQUENCE OF FcFxpPortC1AcctEntry
Access	not-accessible
Status	mandatory
Description	A table that contains, one entry for each FxPort in the Fabric Element, Class 1 accounting information recorded since the management agent has re-initialized.
Sequence	::= { fcFeAcct 1 }

fcFxpPortC1AcctEntry

Syntax	FcFxpPortC1AcctEntry
Access	not-accessible

Status	mandatory
Description	An entry containing Class 1 accounting information for each FxPort. INDEX { fcFxPortC1AcctModuleIndex, fcFxPortC1AcctFxPortIndex }
Sequence	::= { fcFxPortC1AcctTable 1 } FcFxPortC1AcctEntry ::= SEQUENCE { fcFxPortC1AcctModuleIndex FcFeModuleIndex, fcFxPortC1AcctFxPortIndex FcFeFxPortIndex, fcFxPortC1InConnections Counter, fcFxPortC1OutConnections Counter, fcFxPortC1FbsyFrames Counter, fcFxPortC1FrjtFrames Counter, fcFxPortC1ConnTime Counter, fcFxPortC1InFrames Counter, fcFxPortC1OutFrames Counter, fcFxPortC1InOctets Counter, fcFxPortC1OutOctets Counter, fcFxPortC1Discards Counter }

fcFxPortC1AcctModuleIndex

Syntax	FcFeModuleIndex
Access	read-only
Status	mandatory
Description	This object identifies the module containing the FxPort for which this entry contains information.
Sequence	::= { fcFxPortC1AcctEntry 1 }

fcFxPortC1AcctFxPortIndex

Syntax	FcFeFxPortIndex
Access	read-only
Status	mandatory

Description This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.

Sequence ::= { fcFxPortC1AcctEntry 2 }

fcFxPortC1InConnections

SYNTAX Counter

Access read-only

Status mandatory

Description The number of Class 1 connections successfully established in which the attached NxPort is the source of the connect-request.

Sequence ::= { fcFxPortC1AcctEntry 3 }

fcFxPortC1OutConnections

SYNTAX Counter

Access read-only

Status mandatory

Description The number of Class 1 connections successfully established in which the attached NxPort is the destination of the connect-request.

Sequence ::= { fcFxPortC1AcctEntry 4 }

fcFxPortC1FbsyFrames

SYNTAX Counter

Access read-only

Status mandatory

Description The number of F_BSY frames generated by this FxPort against Class 1 connect-request.

Sequence ::= { fcFxPortC1AcctEntry 5 }

fcFxpPortC1FrjtFrames

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of F_RJT frames generated by this FxPort against Class 1 connect-request.
Sequence	::= { fcFxpPortC1AcctEntry 6 }

fcFxpPortC1ConnTime

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The cumulative time that this FxPort has been engaged in Class 1 connection. The amount of time of each connection is counted in octets from after a connect- request has been accepted until the connection is disengaged, either by an EOFdt or Link Reset.
Sequence	::= { fcFxpPortC1AcctEntry 7 }

fcFxpPortC1InFrames

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Class 1 frames (other than Class 1 connect-request) received by this FxPort from its attached NxPort.
Sequence	::= { fcFxpPortC1AcctEntry 8 }

fcFxpPortC1OutFrames

SYNTAX	Counter
Access	read-only
Status	mandatory

Description The number of Class 1 frames (other than Class 1 connect-request) delivered through this FxPort to its attached NxPort.

Sequence ::= { fcFxPortC1AcctEntry 9 }

fcFxPortC1InOctets

SYNTAX Counter

Access read-only

Status mandatory

Description The number of Class 1 frame octets, including the frame delimiters, received by this FxPort from its attached NxPort.

Sequence ::= { fcFxPortC1AcctEntry 10 }

fcFxPortC1OutOctets

SYNTAX Counter

Access read-only

Status mandatory

Description The number of Class 1 frame octets, including the frame delimiters, delivered through this FxPort its attached NxPort.

Sequence ::= { fcFxPortC1AcctEntry 11 }

fcFxPortC1Discards

SYNTAX Counter

Access read-only

Status mandatory

Description The number of Class 1 frames discarded by this FxPort.

Sequence ::= { fcFxPortC1AcctEntry 12 }

Class 2 Accounting table

This table contains, one entry for each FxPort in the Fabric Element, Counters for certain types of events occurred in the the FxPorts since the the management agent has re-initialized. Implementation of this group is optional.

fcFxPortC2AcctTable

Syntax	SEQUENCE OF FcFxPortC2AcctEntry
Access	not-accessible
Status	mandatory
Description	A table that contains, one entry for each FxPort in the Fabric Element, Class 2 accounting information recorded since the management agent has re-initialized.
Sequence	::= { fcFeAcct 2 }

fcFxPortC2AcctEntry

Syntax	FcFxPortC2AcctEntry
Access	not-accessible
Status	mandatory
Description	An entry containing Class 2 accounting information for each FxPort." INDEX { fcFxPortC2AcctModuleIndex, fcFxPortC2AcctFxPortIndex }
Sequence	::= { fcFxPortC2AcctTable 1 } FcFxPortC2AcctEntry ::= SEQUENCE { fcFxPortC2AcctModuleIndex FcFeModuleIndex, fcFxPortC2AcctFxPortIndex FcFeFxPortIndex, fcFxPortC2InFrames Counter, fcFxPortC2OutFrames Counter, fcFxPortC2InOctets Counter, fcFxPortC2OutOctets Counter,

fcFxpPortC2Discards	Counter,
fcFxpPortC2FbsyFrames	Counter,
fcFxpPortC2FrjtFrames	Counter }

fcFxpPortC2AcctModuleIndex

Syntax	FcFeModuleIndex
Access	read-only
Status	mandatory
Description	This object identifies the module containing the FxpPort for which this entry contains information.
Sequence	::= { fcFxpPortC2AcctEntry 1 }

fcFxpPortC2AcctFxpPortIndex

Syntax	FcFeFxpPortIndex
Access	read-only
Status	mandatory
Description	This object identifies the FxpPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxpPort until the module is re-initialized.
Sequence	::= { fcFxpPortC2AcctEntry 2 }

fcFxpPortC2InFrames

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Class 2 frames received by this FxpPort from its attached NxPort.
Sequence	::= { fcFxpPortC2AcctEntry 3 }

fcFxpPortC2OutFrames

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Class 2 frames delivered through this FxPort to its attached NxPort.
Sequence	::= { fcFxpPortC2AcctEntry 4 }

fcFxpPortC2InOctets

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Class 2 frame octets, including the frame delimiters, received by this FxPort from its attached NxPort.
Sequence	::= { fcFxpPortC2AcctEntry 5 }

fcFxpPortC2OutOctets

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Class 2 frame octets, including the frame delimiters, delivered through this FxPort to its attached NxPort.
Sequence	::= { fcFxpPortC2AcctEntry 6 }

fcFxpPortC2Discards

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Class 2 frames discarded by this FxPort.
Sequence	::= { fcFxpPortC2AcctEntry 7 }

fcFxpPortC2FbsyFrames

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of F_BSY frames generated by this FxPort against Class 2 frames.
Sequence	::= { fcFxpPortC2AcctEntry 8 }

fcFxpPortC2FrjtFrames

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of F_RJT frames generated by this FxPort against Class 2 frames.
Sequence	::= { fcFxpPortC2AcctEntry 9 }

Class 3 Accounting Group

This table contains, one entry for each FxPort in the Fabric Element, Counters for certain types of events occurred in the the FxPorts since the management agent has re-initialized. Implementation of this group is optional.

fcFxpPortC3AcctTable

Syntax	SEQUENCE OF FcFxpPortC3AcctEntry
Access	not-accessible
Status	mandatory
Description	A table that contains, one entry for each FxPort in the Fabric Element, Class 3 accounting information recorded since the management agent has re-initialized.
Sequence	::= { fcFeAcct 3 }

fcFxpPortC3AcctEntry

Syntax	FcFxpPortC3AcctEntry
Access	not-accessible
Status	mandatory
Description	An entry containing Class 3 accounting information for each FxPort. INDEX { fcFxpPortC3AcctModuleIndex, fcFxpPortC3AcctFxpPortIndex }
Sequence	::= { fcFxpPortC3AcctTable 1 } FcFxpPortC3AcctEntry ::= SEQUENCE { fcFxpPortC3AcctModuleIndex FcFeModuleIndex, fcFxpPortC3AcctFxpPortIndex FcFeFxpPortIndex, fcFxpPortC3InFrames Counter, fcFxpPortC3OutFrames Counter, fcFxpPortC3InOctets Counter, fcFxpPortC3OutOctets Counter, fcFxpPortC3Discards Counter }

fcFxpPortC3AcctModuleIndex

Syntax	FcFeModuleIndex
Access	read-only
Status	mandatory
Description	This object identifies the module containing the FxPort for which this entry contains information.
Sequence	::= { fcFxpPortC3AcctEntry 1 }

fcFxpPortC3AcctFxpPortIndex

Syntax	FcFeFxpPortIndex
Access	read-only
Status	mandatory

Description This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.

Sequence ::= { fcFxPortC3AcctEntry 2 }

fcFxPortC3InFrames

SYNTAX Counter

Access read-only

Status mandatory

Description The number of Class 3 frames received by this FxPort from its attached NxPort.

Sequence ::= { fcFxPortC3AcctEntry 3 }

fcFxPortC3OutFrames

SYNTAX Counter

Access read-only

Status mandatory

Description The number of Class 3 frames delivered through this FxPort to its attached NxPort.

Sequence ::= { fcFxPortC3AcctEntry 4 }

fcFxPortC3InOctets

SYNTAX Counter

Access read-only

Status mandatory

Description The number of Class 3 frame octets, including the frame delimiters, received by this FxPort from its attached NxPort.

Sequence ::= { fcFxPortC3AcctEntry 5 }

fcFxpPortC3OutOctets

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Class 3 frame octets, including the frame delimiters, delivered through this FxPort to its attached NxPort.
Sequence	::= { fcFxpPortC3AcctEntry 6 }

fcFxpPortC3Discards

SYNTAX	Counter
Access	read-only
Status	mandatory
Description	The number of Class 3 frames discarded by this FxPort.
Sequence	::= { fcFxpPortC3AcctEntry 7 }

Capability Group

The Capability Group - consists of a table describing information about what each FxPort is inherently capable of operating or supporting.

A capability may be used, as expressed in its respective object value in the Configuration group. Implementation of this group is optional.

fcFxpPortCapTable

Syntax	SEQUENCE OF FcFxpPortCapEntry
Access	not-accessible
Status	mandatory
Description	A table that contains, one entry for each FxPort, the capabilities of the port within the Fabric Element.
Sequence	::= { fcFeCap 1 }

fcFxPortCapEntry

Syntax	FcFxPortCapEntry
Access	not-accessible
Status	mandatory
Description	An entry containing the capabilities of a FxPort. INDEX { fcFxPortCapModuleIndex, fcFxPortCapFxPortIndex }
Sequence	::= { fcFxPortCapTable 1 } FcFxPortCapEntry ::= SEQUENCE { fcFxPortCapModuleIndex FcFeModuleIndex, fcFxPortCapFxPortIndex FcFeFxPortIndex, fcFxPortCapFcphVersionHigh FcphVersion, fcFxPortCapFcphVersionLow FcphVersion, fcFxPortCapBbCreditMax FcBbCredit, fcFxPortCapBbCreditMin FcBbCredit, fcFxPortCapRxDataFieldSizeMax FcRxDataFieldSize, fcFxPortCapRxDataFieldSizeMin FcRxDataFieldSize, fcFxPortCapCos FcCosCap, fcFxPortCapIntermix INTEGER, fcFxPortCapStackedConnMode FcStackedConnMode, fcFxPortCapClass2SeqDeliv INTEGER, fcFxPortCapClass3SeqDeliv INTEGER, fcFxPortCapHoldTimeMax MicroSeconds, fcFxPortCapHoldTimeMin MicroSeconds, fcFxPortCapBaudRates Fc0BaudRateCap, fcFxPortCapMedia Fc0MediaCap }

fcFxPortCapModuleIndex

Syntax	FcFeModuleIndex
Access	read-only

Status	mandatory
Description	This object identifies the module containing the FxPort for which this entry contains information.
Sequence	::= { fcFxPortCapEntry 1 }

fcFxPortCapFxPortIndex

Syntax	FcFeFxPortIndex
Access	read-only
Status	mandatory
Description	This object identifies the FxPort within the module. This number ranges from 1 to the value of fcFeModulePortCapacity for the associated module. The value remains constant for the identified FxPort until the module is re-initialized.
Sequence	::= { fcFxPortCapEntry 2 }

fcFxPortCapFcphVersionHigh

Syntax	FcphVersion
Access	read-only
Status	mandatory
Description	The highest or most recent version of FC-PH that the FxPort is capable of supporting.
Sequence	::= { fcFxPortCapEntry 3 }

fcFxPortCapFcphVersionLow

Syntax	FcphVersion
Access	read-only
Status	mandatory
Description	The lowest or earliest version of FC-PH that the FxPort is capable of supporting.
Sequence	::= { fcFxPortCapEntry 4 }

fcFxpPortCapBbCreditMax

Syntax	FcBbCredit
Access	read-only
Status	mandatory
Description	The maximum number of receive buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames from the attached NxPort.
Sequence	::= { fcFxpPortCapEntry 5 }

fcFxpPortCapBbCreditMin

Syntax	FcBbCredit
Access	read-only
Status	mandatory
Description	The minimum number of receive buffers available for holding Class 1 connect-request, Class 2 or Class 3 frames from the attached NxPort.
Sequence	::= { fcFxpPortCapEntry 6 }

fcFxpPortCapRxDataFieldSizeMax

Syntax	FcRxDataFieldSize
Access	read-only
Status	mandatory
Description	The maximum size in bytes of the Data Field in a frame that the FxPort is capable of receiving from its attached NxPort.
Sequence	::= { fcFxpPortCapEntry 7 }

fcFxpPortCapRxDataFieldSizeMin

Syntax	FcRxDataFieldSize
Access	read-only
Status	mandatory
Description	The minimum size in bytes of the Data Field in a frame that the FxPort is capable of receiving from its attached NxPort.

Sequence ::= { fcFxpPortCapEntry 8 }

fcFxpPortCapCos

Syntax FcCosCap
 Access read-only
 Status mandatory
 Description A value indicating the set of Classes of Service that the FxpPort is capable of supporting.
 Sequence ::= { fcFxpPortCapEntry 9 }

fcFxpPortCapIntermix

Syntax INTEGER { yes (1), no (2) }
 Access read-only
 Status mandatory
 Description A flag indicating whether or not the FxpPort is capable of supporting the intermixing of Class 2 and Class 3 frames during a Class 1 connection. This flag is only valid if the port is capable of supporting Class 1 service.
 Sequence ::= { fcFxpPortCapEntry 10 }

fcFxpPortCapStackedConnMode

Syntax FcStackedConnMode
 Access read-only
 Status mandatory
 Description A value indicating the mode of Stacked Connect request that the FxpPort is capable of supporting.
 Sequence ::= { fcFxpPortCapEntry 11 }

fcFxpPortCapClass2SeqDeliv

Syntax INTEGER { yes (1), no (2) }
 Access read-only

Status	mandatory
Description	A flag indicating whether or not the FxPort is capable of supporting Class 2 Sequential Delivery.
Sequence	::= { fcFxPortCapEntry 12 }

fcFxPortCapClass3SeqDeliv

Syntax	INTEGER { yes (1), no (2) }
Access	read-only
Status	mandatory
Description	A flag indicating whether or not the FxPort is capable of supporting Class 3 Sequential Delivery.
Sequence	::= { fcFxPortCapEntry 13 }

fcFxPortCapHoldTimeMax

Syntax	MicroSeconds
Access	read-only
Status	mandatory
Description	The maximum holding time (in microseconds) that the FxPort is capable of supporting.
Sequence	::= { fcFxPortCapEntry 14 }

fcFxPortCapHoldTimeMin

Syntax	MicroSeconds
Access	read-only
Status	mandatory
Description	The minimum holding time (in microseconds) that the FxPort is capable of supporting.
Sequence	::= { fcFxPortCapEntry 15 }

fcFxpPortCapBaudRates

Syntax	Fc0BaudRateCap
Access	read-only
Status	deprecated
Description	A value indicating the set of baud rates that the FxPort is capable of supporting. This variable has been deprecated and may be implemented for backward compability.
Sequence	::= { fcFxpPortCapEntry 16 }

fcFxpPortCapMedia

Syntax	Fc0MediaCap
Access	read-only
Status	deprecated
Description	A value indicating the set of media that the FxPort is capable of supporting. This variable has been deprecated and may be implemented for backward compability.
Sequence	::= { fcFxpPortCapEntry 17 } END

A

agents, introduction to [1-1](#)

C

commands, SNMP [1-2](#)

E

EOS trap overview [2-3](#)

F

fabric element management MIB
 class 1 accounting table [2-85](#)
 class 2 accounting table [2-88](#)
 class 3 accounting table [2-90](#)
 description of [2-61](#)
 FxPort capability table [2-91](#)
 FxPort configuration table [2-68](#)
 FxPort error table [2-82](#)
 FxPort fabric login table [2-78](#)
 FxPort operation table [2-74](#)
 FxPort physical level table [2-76](#)
 module table [2-66](#)
 objects defined in [2-66](#)
 predefined types [2-61](#)
Fc_Port [2-13](#)
FCEOS MIB
 enterprise specific traps [2-12](#)
fcEosPortScn [2-13](#)
Fibre Alliance MIB [1-4](#)
fibre alliance MIB (FA MIB)
 connectivity unit group [2-97](#)
 description of [2-96](#)

event table [2-119](#)
fcConnUnitTable [2-98](#)
firmware table [2-106](#)
link table [2-122](#)
name server table [2-141](#)
port statistics [2-126](#)
port table [2-109](#)
sensor table [2-107](#)
SNMP trap registration group [2-144](#)
trap registration table [2-145](#)
trap types [2-146](#)
type definitions [2-96](#)
Fibre Channel Fabric Element (FCFE) MIB [1-5](#)

G

getnextrequest command [1-2](#)
getrequest command [1-2](#)
getresponse command [1-3](#)

M

McDATA private enterprise MIBs [1-5](#)
MIB II
 IP Routing table [2-37](#)
MIB objects
 table description
 atIfIndex [2-30](#)
 atNetAddress [2-31](#)
 atPhysAddress [2-30](#)
 FabricName [2-61](#)
 Fc0BaudRate [2-63](#)
 Fc0BaudRateCap [2-63](#)
 Fc0Distance [2-65](#)
 Fc0MediaCap [2-64](#)

- Fc0Medium [2-64](#)
- Fc0TxType [2-64](#)
- FcAddressId [2-62](#)
- FcBbCredit [2-62](#)
- FcBbCreditModel [2-66](#)
- FcConnUnitContact [2-105](#)
- FcConnUnitControl [2-104](#)
- fcConnUnitDeletedTrap [2-147](#)
- FcConnUnitDomainId [2-102](#)
- FcConnUnitEventCurrID [2-106](#)
- fcConnUnitEventDescr [2-122](#)
- FcConnUnitEventFilter [2-105](#)
- fcConnUnitEventIndex [2-119](#)
- FcConnUnitEventObject [2-121](#)
- FcConnUnitEventSeverity [2-120](#)
- fcConnUnitEventTrap [2-147](#)
- FcConnUnitEventType [2-121](#)
- FcConnUnitGlobalId [2-99](#)
- fcConnUnitId [2-98](#)
- FcConnUnitInfo [2-104](#)
- FcConnUnitLinkAgentAddressTypeY [2-125](#)
- FcConnUnitLinkAgentAddressY [2-125](#)
- FcConnUnitLinkAgentPortY [2-125](#)
- FcConnUnitLinkConnIdY [2-125](#)
- fcConnUnitLinkIndex [2-123](#)
- FcConnUnitLinkNodeIdx [2-123](#)
- FcConnUnitLinkNodeIdY [2-124](#)
- FcConnUnitLinkPortNumberX [2-123](#)
- FcConnUnitLinkPortNumberY [2-124](#)
- FcConnUnitLinkPortWwnX [2-124](#)
- FcConnUnitLinkPortWwnY [2-124](#)
- FcConnUnitLinkUnitTypeY [2-125](#)
- FcConnUnitLocation [2-105](#)
- FcConnUnitMaxEvents [2-106](#)
- FcConnUnitModuleId [2-103](#)
- FcConnUnitName [2-103](#)
- fcConnUnitNumber [2-97](#)
- FcConnUnitNumEvents [2-106](#)
- FcConnUnitNumports [2-100](#)
- FcConnUnitNumRevs [2-103](#)
- FcConnUnitNumSensors [2-102](#)
- FcConnUnitPortControl [2-115](#)
- FcConnUnitPortFCClassCap [2-111](#)
- FcConnUnitPortFCClassOp [2-111](#)
- FcConnUnitPortFCId [2-114](#)
- FcConnUnitPortHWState [2-118](#)
- fcConnUnitPortIndex [2-109](#)
- fcConnUnitPortModuleType [2-113](#)
- FcConnUnitPortName [2-116](#)
- FcConnUnitPortNodeWwn [2-118](#)
- FcConnUnitPortPhysicalNumber [2-116](#)
- FcConnUnitPortProtocolCap [2-117](#)
- FcConnUnitPortProtocolOp [2-117](#)
- FcConnUnitPortRevision [2-114](#)
- FcConnUnitPortSerialNoSn [2-114](#)
- FcConnUnitPortSpeed [2-115](#)
- fcConnUnitPortStatCountAddressErrors [2-140](#)
- FcConnUnitPortStatCountBBCreditZero [2-128](#)
- FcConnUnitPortStatCountClass1FBSYFrames [2-130](#)
- FcConnUnitPortStatCountClass1FRJTFrames [2-131](#)
- FcConnUnitPortStatCountClass1PBSYFrames [2-131](#)
- FcConnUnitPortStatCountClass1PRJTFrames [2-131](#)
- FcConnUnitPortStatCountClass1RxFrames [2-130](#)
- FcConnUnitPortStatCountClass1TxFrames [2-130](#)
- FcConnUnitPortStatCountClass2FBSYFrames [2-132](#)
- FcConnUnitPortStatCountClass2FRJTFrames [2-133](#)
- FcConnUnitPortStatCountClass2PBSYFrames [2-133](#)
- FcConnUnitPortStatCountClass2PRJTFrames [2-133](#)
- FcConnUnitPortStatCountClass2RxFrames [2-132](#)
- FcConnUnitPortStatCountClass2TxFrames [2-132](#)
- FcConnUnitPortStatCountClass3Discards [2-134](#)
- FcConnUnitPortStatCountClass3RxFrames [2-134](#)
- FcConnUnitPortStatCountClass3TxFrames [2-134](#)
- FcConnUnitPortStatCountDelimiterErrors [2-140](#)
- FcConnUnitPortStatCountEncodingDis-

- parityErrors 2-141
- fcConnUnitPortStatCountError 2-126
- FcConnUnitPortStatCountFBSYFrames 2-129
- FcConnUnitPortStatCountFramesTooLong 2-139
- FcConnUnitPortStatCountFramesTruncated 2-140
- FcConnUnitPortStatCountInputBuffers-Full 2-128
- FcConnUnitPortStatCountInvalidCRC 2-137
- FcConnUnitPortStatCountInvalidOrderedSets 2-139
- FcConnUnitPortStatCountInvalidTx-Words 2-138
- FcConnUnitPortStatCountLinkFailures 2-137
- FcConnUnitPortStatCountLossofSignal 2-138
- FcConnUnitPortStatCountLossofSynchronization 2-139
- FcConnUnitPortStatCountNumber-LinkResets 2-136
- FcConnUnitPortStatCountNumberOf-flineSequences 2-137
- FcConnUnitPortStatCountPBSYFrames 2-129
- FcConnUnitPortStatCountPrimitiveSequenceProtocolErrors 2-138
- FcConnUnitPortStatCountPRJTFrames 2-130
- FcConnUnitPortStatCountRxBroadcastObjects 2-135
- FcConnUnitPortStatCountRxElements 2-128
- FcConnUnitPortStatCountRxLinkResets 2-135
- FcConnUnitPortStatCountRxMulticastObjects 2-134
- FcConnUnitPortStatCountRxObjects 2-127
- FcConnUnitPortStatCountRxOfflineSequences 2-136
- FcConnUnitPortStatCountTxBroadcastObjects 2-135
- FcConnUnitPortStatCountTxElements 2-127
- FcConnUnitPortStatCountTxLinkResets 2-136
- FcConnUnitPortStatCountTxMulticastObjects 2-135
- FcConnUnitPortStatCountTxObjects 2-127
- FcConnUnitPortStatCountTxOfflineSequences 2-136
- FcConnUnitPortState 2-112
- fcConnUnitPortStatIndex 2-126
- FcConnUnitPortStatisticCountFRJT-Frames 2-129
- FcConnUnitPortStatus 2-112
- fcConnUnitPortStatusChange 2-147
- FcConnUnitPortTransmitterType 2-113
- FcConnUnitPortType 2-110
- FcConnUnitPortVendor 2-114
- FcConnUnitPortWwn 2-113
- FcConnUnitPrincipal 2-102
- fcConnUnitProduct 2-101
- FcConnUnitProxyMaster 2-102
- FcConnUnitREventTime 2-120
- FcConnUnitRevsDescription 2-107
- fcConnUnitRevsIndex 2-106
- FcConnUnitRevsRevision 2-107
- FcConnUnitSensorCharacteristic 2-109
- fcConnUnitSensorIndex 2-107
- FcConnUnitSensorInfo 2-108
- FcConnUnitSensorMessage 2-108
- FcConnUnitSensorName 2-108
- FcConnUnitSensorStatus 2-108
- fcConnUnitSensorStatusChange 2-147
- FcConnUnitSensorType 2-109
- FcConnUnitSerialNo 2-101
- FcConnUnitSEventTime 2-120
- fcConnUnitSnsClassOfSvc 2-142
- fcConnUnitSnsFabricPortName 2-143
- fcConnUnitSnsFC4Type 2-143
- fcConnUnitSnsHardAddress 2-144
- FcConnUnitSnsMaxRows 2-98
- fcConnUnitSnsNodeIPAddress 2-142
- fcConnUnitSnsNodeName 2-142
- fcConnUnitSnsPortIdentifier 2-141
- fcConnUnitSnsPortIndex 2-141
- fcConnUnitSnsPortIPAddress 2-143
- fcConnUnitSnsPortName 2-141

- fcConnUnitSnsPortType 2-143
- fcConnUnitSnsProcAssoc 2-142
- fcConnUnitSnsSymbolicPortName 2-144
- FcConnUnitState 2-100
- fcConnUnitStatusChange 2-146
- FcConnUnitType 2-100
- FcConnUnitUpTime 2-101
- FcConnUnitUrl 2-101
- FcConnURL 2-97
- FcCosCap 2-62
- FcElementName 2-66
- fcEosFruScn 2-13
- fcEosPortBindingViolation 2-13
- fcEosPortScn 2-12
- fcEosThresholdAlert 2-13
- FcEventSeverity 2-96
- fcFabricName 2-66
- FcFeFxpPortCapacity 2-65
- FcFeFxpPortIndex 2-65
- FcFeModuleCapacity 2-65, 2-66
- fcFeModuleDescr 2-67
- fcFeModuleFxpPortCapacity 2-68
- FcFeModuleIndex 2-65
- FcFeModuleLastChange 2-68
- fcFeModuleName 2-68
- FcFeModuleObjectID 2-67
- fcFeModuleOperStatus 2-67
- FcFeNxPortIndex 2-65
- fcFPortAttachedPortName 2-75
- FcFPortConnectedPort 2-75
- fcFxpConfFxpPortIndex 2-69
- fcFxplogiFxpPortIndex 2-78
- FcFxplogiNxPortIndex 2-78
- FcFxpPortAddressIdErrors 2-84
- FcFxpPortAdminMode 2-76
- FcFxpPortBaudRate 2-73
- FcFxpPortBbCredit 2-70
- FcFxpPortBbCreditAvailable 2-75
- fcFxpPortBbCreditModel 2-81
- fcFxpPortC1AcctFxpPortIndex 2-85
- FcFxpPortC1ConnTime 2-86
- FcFxpPortC1Discards 2-87
- FcFxpPortC1FbsyFrames 2-85
- FcFxpPortC1FrjtFrames 2-86
- FcFxpPortC1InConnections 2-85
- FcFxpPortC1InFrames 2-86
- FcFxpPortC1InOctets 2-87
- FcFxpPortC1OutConnections 2-85
- FcFxpPortC1OutFrames 2-87
- FcFxpPortC1OutOctets 2-87
- fcFxpPortC2AcctFxpPortIndex 2-88
- FcFxpPortC2Discards 2-89
- FcFxpPortC2FbsyFrames 2-89
- FcFxpPortC2FrjtFrames 2-89
- FcFxpPortC2InFrames 2-88
- FcFxpPortC2InOctets 2-88
- FcFxpPortC2OutFrames 2-88
- FcFxpPortC2OutOctets 2-89
- fcFxpPortC3AcctFxpPortIndex 2-90
- FcFxpPortC3Discards 2-91
- FcFxpPortC3InFrames 2-90
- FcFxpPortC3InOctets 2-90
- FcFxpPortC3OutFrames 2-90
- FcFxpPortC3OutOctets 2-91
- FcFxpPortCapBaudRates 2-95
- FcFxpPortCapBbCreditMax 2-92
- FcFxpPortCapBbCreditMin 2-92
- FcFxpPortCapClass2SeqDeliv 2-94
- FcFxpPortCapClass3SeqDeliv 2-94
- FcFxpPortCapCos 2-93
- FcFxpPortCapFcphVersionHigh 2-91
- FcFxpPortCapFcphVersionLow 2-92
- fcFxpPortCapFxpPortIndex 2-91
- FcFxpPortCapHoldTimeMax 2-94
- FcFxpPortCapHoldTimeMin 2-95
- fcFxpPortCapIntermix 2-93
- FcFxpPortCapMedia 2-95
- FcFxpPortCapRxDataFieldSizeMax 2-92
- FcFxpPortCapRxDataFieldSizeMin 2-93
- FcFxpPortCapStackedConnMode 2-93
- FcFxpPortClass2SeqDeliv 2-72
- FcFxpPortClass2SeqDelivAgreed 2-80
- FcFxpPortClass3SeqDeliv 2-72
- FcFxpPortClass3SeqDelivAgreed 2-81
- FcFxpPortConnectedNxPort 2-81
- FcFxpPortCosSuppAgreed 2-79
- FcFxpPortCosSupported 2-71
- FcFxpPortDelimiterErrors 2-83
- FcFxpPortDistance 2-74
- FcFxpPortEdtov 2-71
- fcFxpPortErrorFxpPortIndex 2-82
- FcFxpPortFcphVersionAgreed 2-79
- FcFxpPortFcphVersionHigh 2-69

- FcFxFPortFcphVersionLow 2-70
- FcFxFPortHoldTime 2-72
- FcFxFPortID 2-74
- FcFxFPortIntermixSuppAgreed 2-80
- fcFxFPortIntermixSupported 2-71
- FcFxFPortInvalidCrcs 2-83
- FcFxFPortInvalidTxWords 2-83
- FcFxFPortLinkFailures 2-82
- FcFxFPortLinkResetIns 2-84
- FcFxFPortLinkResetOuts 2-84
- FcFxFPortMedium 2-73
- FcFxFPortMode 2-66
- FcFxFPortName 2-69
- FcFxFPortNxPortBbCredit 2-79
- FcFxFPortNxPortName 2-81
- FcFxFPortNxPortRxDataFieldSize 2-79
- FcFxFPortOlsIns 2-84
- FcFxFPortOlsOuts 2-84
- fcFxFPortOperFxFPortIndex 2-74
- FcFxFPortOperMode 2-76
- FcFxFPortPhysAdminStatus 2-76
- fcFxFPortPhysFxFPortIndex 2-76
- FcFxFPortPhysLastChange 2-77
- FcFxFPortPhysOperStatus 2-77
- FcFxFPortPhysRttov 2-78
- FcFxFPortPrimSeqProtoErrors 2-83
- FcFxFPortRatov 2-70
- FcFxFPortRxBufSize 2-70
- FcFxFPortSigLosses 2-83
- FcFxFPortStackedConnMode 2-72
- FcFxFPortStackedConnModeAgreed 2-80
- FcFxFPortSyncLosses 2-82
- FcFxFPortTxType 2-73
- FcGlobalId 2-96
- FcNameId 2-61, 2-96
- FcphVersion 2-62
- FcPortFCClass 2-97
- FcPortName 2-62
- FcRxDataFieldSize 2-62
- FcStackedConnMode 2-62
- FcTrapClientCount 2-144
- fcTrapMaxClients 2-144
- FcTrapRegFilter 2-145
- fcTrapRegIpAddress 2-145
- fcTrapRegPort 2-145
- FcTrapRegRowState 2-145
- FcUnitType 2-96
- icmpInAddrMaskReps 2-45
- icmpInAddrMasks 2-45
- icmpInDestUnreaches 2-43
- icmpInEchoReps 2-44
- icmpInEchos 2-44
- icmpInErrors 2-43
- icmpInMsgs 2-43
- icmpInParmProbs 2-43
- icmpInRedirects 2-44
- icmpInSrcQuenchs 2-44
- icmpInTimeExcds 2-43
- icmpInTimestampReps 2-44
- icmpInTimestamps 2-44
- icmpOutAddrMaskReps 2-47
- icmpOutAddrMasks 2-47
- icmpOutDestUnreaches 2-45
- icmpOutEchoReps 2-47
- icmpOutEchos 2-46
- icmpOutErrors 2-45
- icmpOutMsgs 2-45
- icmpOutParmProbs 2-46
- icmpOutRedirects 2-46
- icmpOutSrcQuenchs 2-46
- icmpOutTimeExcds 2-46
- icmpOutTimestampReps 2-47
- ifAdminStatus 2-26
- ifDescr 2-24
- ifIndex 2-24
- ifInDiscards 2-28
- ifInErrors 2-28
- ifInNUcastPkts 2-27
- ifInOctets 2-27
- ifInUcastPkts 2-27
- ifInUnknownProtos 2-28
- ifLastChange 2-27
- ifMtu 2-26
- ifNumber 2-23
- ifOperStatus 2-27
- ifOutDiscards 2-29
- ifOutErrors 2-29
- ifOutNUcastPkts 2-29
- ifOutOctets 2-28
- ifOutQLen 2-29
- ifOutUcastPkts 2-28
- ifPhysAddress 2-26
- ifSpecific 2-29
- ifSpeed 2-26

- ipAdEntAddr 2-36
- ipAdEntBcastAddr 2-36
- ipAdEntIfIndex 2-36
- ipAdEntNetMask 2-36
- ipAdEntReasmMaxSize 2-37
- ipDefaultTTL 2-31
- ipForwarding 2-31
- ipForwDatagrams 2-32
- ipFragCreates 2-35
- ipFragFails 2-35
- ipFragOKs 2-35
- ipInAddrErrors 2-32
- ipInDelivers 2-33
- ipInDiscards 2-33
- ipInHdrErrors 2-32
- ipInReceives 2-32
- ipInUnknownProtos 2-33
- ipNetToMediaIfIndex 2-41
- ipNetToMediaNetAddress 2-42
- ipNetToMediaPhysAddress 2-41
- ipNetToMediaType 2-42
- ipOutDiscards 2-33
- ipOutNoRoutes 2-34
- ipOutRequests 2-33
- ipReasmFails 2-35
- ipReasmOKs 2-34
- ipReasmReqds 2-34
- ipReasmTimeout 2-34
- ipRouteAge 2-40
- ipRouteDest 2-37
- ipRouteIfIndex 2-37
- ipRouteInfo 2-41
- ipRouteMask 2-40
- ipRouteMetric1 2-37
- ipRouteMetric2 2-38
- ipRouteMetric3 2-38
- ipRouteMetric4 2-38
- ipRouteMetric5 2-41
- ipRouteNextHop 2-38
- ipRouteProto 2-39
- ipRouteType 2-39
- ipRoutingDiscards 2-42
- snmpEnableAuthenTraps 2-60
- snmpInASNParseErrors 2-55
- snmpInBadCommunityNames 2-55
- snmpInBadCommunityUses 2-55
- snmpInBadValues 2-56
- snmpInBadVersions 2-55
- snmpInGenErrors 2-57
- snmpInGetNexts 2-57
- snmpInGetRequests 2-57
- snmpInGetResponses 2-58
- snmpInNoSuchNames 2-56
- snmpInPkts 2-54
- snmpInReadOnlys 2-56
- snmpInSetRequests 2-58
- snmpInTooBig 2-56
- snmpInTotalReqVars 2-57
- snmpInTotalSetVars 2-57
- snmpInTraps 2-58
- snmpOutBadValues 2-59
- snmpOutGenErrors 2-59
- snmpOutGetNexts 2-59
- snmpOutGetRequests 2-59
- snmpOutGetResponses 2-60
- snmpOutNoSuchNames 2-58
- snmpOutPkts 2-55
- snmpOutSetRequests 2-59
- snmpOutTooBig 2-58
- snmpOutTraps 2-60
- sysContact 2-22
- sysDescr 2-21
- sysLocation 2-22
- sysName 2-22
- sysObjectID 2-21
- sysServices 2-23
- sysUpTime 2-22
- tcpActiveOpens 2-49
- tcpAttemptFails 2-49
- tcpConnLocalAddress 2-51
- tcpConnLocalPort 2-52
- tcpConnRemAddress 2-52
- tcpConnRemPort 2-52
- tcpConnState 2-51
- tcpCurrEstab 2-50
- tcpEstabResets 2-49
- tcpInErrors 2-52
- tcpInSegs 2-50
- tcpMaxConn 2-49
- tcpOutRsts 2-52
- tcpOutSegs 2-50
- tcpPassiveOpens 2-49
- tcpRetransSegs 2-50
- tcpRtoAlgorithm 2-48

- tcpRtoMax 2-48
- tcpRtoMin 2-48
- udpInDatagrams 2-53
- udpInErrors 2-53
- udpLocalAddress 2-54
- udpLocalPort 2-54
- udpNoPorts 2-53
- udpOutDatagrams 2-53

MIB-II

- additional IP objects 2-42
- additional TCP objects 2-52
- address translation group 2-30
- definition of 2-21
- ICMP group 2-43
- interfaces group 2-23
- IP address tabler 2-36
- IP address translation table 2-41
- IP group 2-31
- SNMP group 2-54
- system group 2-21
- TCP connection table 2-51
- TCP group 2-48
- UDP group 2-53
- UDP listener table 2-54

MIBs

- Fibre Alliance MIB 1-4
- Fibre Channel Fabric Element (FCFE) MIB 1-5
- introduction to 1-4
- MIB-II 1-4
- MIB-II, definition of 2-21
- standard 1-4

MIBs support by McDATA 1-4

N

network management, introduction to 1-1

P

- port binding
 - violation 2-13
- private enterprise MIBs 1-5

S

- setrequest command 1-3
- SNMP

- description of 1-2
- introduction to 1-1
- SNMP commands 1-2
- standard MIBs 1-4

T

- trap information, interpretation of (HP OpenView) 2-17
- trap information, interpretation of (MG-SOFT MIB browser) 2-19
- trap PDU, format of 2-4
- traps
 - a new connection has been established on a port 2-17
 - bit error rate for a link exceeds the threshold 2-15
 - definition of 1-5
 - Fibre Channel port operational state change 2-12
 - field descriptions 2-3
 - FRU operational state change 2-13
 - FRU removed or status unknown 2-13
 - FRU status changes to active 2-14
 - FRU status changes to backup 2-14
 - FRU status changes to failed 2-15
 - FRU status changes to update/busy 2-14
 - invalid primitive sequence 2-16
 - loss of signal or sync 2-15
 - message format 2-3
 - not operational primitive sequence is received 2-16
 - overview, EOS 2-3
 - PDU format 2-3
 - Port binding violation 2-13
 - primitive sequence timeout 2-16
 - purpose of 1-5
 - Threshold alert 2-13
 - UDP packets 2-3

U

- UDP packets containing traps 2-3

V

- variables
 - how SNMP changes 1-3

introduction to [1-3](#)
violation [2-13](#)